Generalized Discriminative Training for Speech Recognition

Roger Hsiao

CMU-LTI-12-001

Jan 5, 2012

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Tanja Schultz (Chair)
Alan Black
Florian Metze
George Saon, IBM T.J. Watson Research Center

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Keywords: Speech recognition, discriminative training

Soli Deo Gloria

Abstract

Discriminative training for speech recognition aims to minimize the errors caused by the generative models. It is often formulated as an optimization problem involving the references and the competing hypotheses. While discriminative training can improve recognition performance, it comes with a few drawbacks. First, the optimization problem is difficult to solve due to the complex objective functions. This leads to the need of heuristics and smoothing techniques for optimization. Second, discriminative training is time consuming since it can take days or weeks to finish on large systems.

The goal of this thesis is to reformulate the optimization problems of discriminative training, so that we can develop better optimization algorithms which are more efficient. Our methods are based on Lagrange relaxation which we convert the difficult optimization problems into simpler convex problems. Our proposed generalized Baum-Welch (GBW) algorithm is a generalization of the Baum-Welch (BW) algorithm and the extended Baum-Welch (EBW) algorithm. Through the GBW framework, we discover an interesting connection between EBW and information theory. This inspires us to develop better and faster EBW variants, including the recursive EBW algorithm and statistical EBW algorithm.

By using the same framework of GBW, we propose generalized discriminative feature transformation (GDFT) algorithm which transforms the constrained maximum likelihood regression (CMLLR) to perform feature space discriminative training. We compare our GDFT with the state of the art feature space maximum mutual information (fMMI), and show that GDFT is competitive in accuracy and runs much faster.

Based on our proposed algorithms, we introduce single pass discriminative training which aims to extract as much improvement as possible by only allowing process the data once. Our experiments show that single pass training can obtain 80-90

Abbreviations

ASR	Automatic Speech Recognition
BW	Baum-Welch
BMMI	Boosted Maximum Mutual Information
CMLLR	Constrained Maximum Likelihood Linear Regression
DFE	Discriminative Feature Extraction
DFT	Discriminative Feature Transformation
EBW	Extended Baum-Welch
EM	Expectation Maximization
fMPE	Feature Minimum Phone Error
fMMI	Feature Maximum Mutual Information
GALE	Global Autonomous Language Exploitation
GBW	Generalized Baum-Welch
GDFT	Generalized Discriminative Feature Transformation
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
LVCSR	Large Vocabulary Continuous Speech Recognition
MCE	Minimum Classification Error
ML	Maximum Likelihood
MLLR	Maximum Likelihood Linear Regression
MMI	Maximum Mutual Information
MPE	Minimum Phone Error
RDFT	Region Dependent Feature Transformation
RDLT	Region Dependent Linear Transformation
rEBW	Recursive Extended Baum-Welch
SAT	Speaker Adaptive Training
sEBW	Statistical Extended Baum-Welch
TransTac	Spoken Language Communication and Translation System for Tactical Use
VTLN	Vocal Tract Length Normalization
WER	Word Error Rate

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Discriminative training is one of the major topics in speech recognition research. Successful applications of discriminative training, especially the improvement to the large scale systems, have continued to draw a lot of attention to researchers in the past 20 years.

The fundamental assumption of discriminative training is based on the imperfectness of the models, which causes the conventional maximum likelihood (ML) approaches to be suboptimal in terms of classification accuracy. In the case of speech recognition, the acoustic model, i.e. the hidden Markov model (HMM), is known to be incorrect in many ways for modeling human speech. For instance, the first order assumption and the independent output assumption, are desirable from the computational and statistical point of view, but they are not realistic to the data that we would like to model. Hence, the ML approach may not give the best performance, and researchers have been studying alternative model parameter estimation techniques like discriminative training.

In general, discriminative training can be roughly divided into three parts: the recognition error function, the optimization algorithm and the model that receives optimization. The goal of discriminative training is to optimize the model parameters such that the recognition error is minimized on the train data.

In discriminative training, the recognition error is often expressed as different forms of objective functions that involve the reference and the competing hypotheses. These ob-

jective functions can be considered as some smoothed versions of word error rate (WER) which are suitable for optimization. Notable discriminative training examples include, but not limited to, maximum mutual information (MMI) [Valtchev et al. (1997)], minimum phone error (MPE) [Povey (2003)] and minimum classification error (MCE) [Juang and Katagiri (1992a)]. These discriminative training algorithms optimize the HMM parameters for their smoothed recognition error functions (mutual information, phone error). These functions are often more complicated than the log likelihood function used in the ML approach. Thus, the optimization procedure are also more time consuming and often requires careful tuning. While early discriminative training research focuses on the HMM, researchers later investigated the possibility of applying discriminative training on other areas like feature extraction [Biem et al. (2001) ; Mak et al. (2002)], feature transformation [Povey et al. (2005) ; Povey (2005) ; Zhang et al. (2006a) ; Zhang et al. (2006b)], speaker adaptation [Gunawardana and Byrne (2001) ; Wang and Woodland (2004)], and unsupervised training [Yu et al. (2007)]. Encouraging results have been reported and driven the research on discriminative training.

While discriminative training is useful to improve speech recognition performance, it comes with a few drawbacks. As mentioned, due to the complicated objective functions, the optimization is difficult and the existing optimization algorithms often involves a lot of heuristics and tuning. Another drawback of discriminative training is the very long training time, since in addition to estimating the parameters based on the references, discriminative training also needs to consider the competing hypotheses. As a result, the practice and the implementation of discriminative training is often considered to be difficult and challenging.

The goal of this thesis is to propose a family of optimization algorithms which are simple and efficient. When tuning and using some heuristics become necessary, the theories behind the algorithms should explain the meaning of the tuning parameters, and give the users some basic ideas about how to tune properly instead of using a pure empirical approach.

1.1 Optimization for Discriminative Training

This section aims to provide a brief introduction about the optimization problem of discriminative training for acoustic modeling, and explain why such optimization is difficult. A more detailed discussion is available in chapter 2.

Discriminative training is often formulated as an optimization problem which targets at minimizing the recognition error. Although word error rate (WER) is the target function to be minimized, the function is non-differentiable and not smooth which makes direct optimization difficult. Instead, discriminative training optimizes a smoothed approximation of WER. One possible choice is the maximum mutual information(MMI). Consider

$$f(X,\theta) \quad = \quad \log \frac{P(X|W;\theta)P(W)}{\sum_{W'} P(X|W';\theta)P(W')} \tag{1.1}$$

where $X \equiv x_1, x_2, \ldots, x_T$ is an observation sequence with T frames; θ represents the set of model parameters to be optimized; W is the reference word sequence for X; The denominator represents it considers all possible word sequences as the competing hypotheses. Maximizing the function f is the same as maximizing the empirical mutual information between X and W. Intuitively speaking, this objective function aims to keep the likelihood of the reference intact and at the same time, reduce the likelihood of the competing hypotheses. As a result, the optimized model will be less likely to be confused by the wrong hypotheses and have a better chance to perform recognition correctly.

Optimization of f is not trivial. Assuming θ refers to the HMM parameters which include the Gaussian means and covariances, the objective function is not concave with respect to θ. As a result, the solution from optimizing f is likely to be local optimal. Another difficulty of this optimization problem is the unbounded issue. Assuming there is one Gaussian which only appears in the denominator. The optimization problem for this particular Gaussian would become a minimum likelihood problem. The solution of the minimum likelihood problem is not bounded because to keep the likelihood zero, either the mean of the Gaussian has to be infinitely far away from the input feature or the covariance has to be zero, which are undesirable in both cases. In general, if the occupation count of a Gaussian as a competing hypothesis is higher than its occupation count as a reference, it

triggers the unbounded issue, and therefore, the optimization is difficult.

1.2 Proposed Research

The goal of this thesis is to propose a family of optimization algorithms which are simple and efficient. When tuning and heuristics become necessary, the theories behind the algorithms should explain the meaning of the parameters, and give the users some basic ideas about tuning. Therefore, we reformulate the optimization problem for discriminative training, and propose new optimization algorithms based on Lagrange relaxation [Boyd and Vandenberghe (2004)]. In which, we relax the difficult optimization problems into simpler convex problems. We propose the generalized Baum-Welch (GBW) algorithm for model space discriminative training, and the generalized discriminative feature transformation (GDFT) for feature space discriminative training. The GBW algorithm generalizes the Baum-Welch (BW) and the extended Baum-Welch (EBW) algorithm for HMM. The GBW formulation shows the heuristics and the smoothing techniques used by the EBW algorithm can be expressed as some distance based regularization in the optimization problem. This formulation also reveals interesting connection between the EBW algorithm and information theory, and inspires better EBW variants. Based on the GBW framework, the GDFT algorithm transforms the constrained maximum likelihood regression (CMLLR) algorithm to perform feature space discriminative training. Its formulation shows there are efficient ways to combine model space and feature space discriminative training.

1.3 Thesis Organization

In chapter 2, we discuss the existing methods for discriminative training which include the objective functions and the optimization algorithms for HMM. We also compare some of the feature space discriminative training algorithms to date. We describe the baseline ASR systems used and the data sets for experiments in chapter 3. In chapter 4, we propose the GBW algorithm for model space discriminative training and we show that both Baum-

Welch (BW) algorithm and the extended Baum-Welch (EBW) algorithm are special cases of GBW. In addition, we show how the GBW formulation can lead to better variants of the EBW algorithm. Chapter 5 is about our feature space discriminative training algorithm, GDFT. We explore how GDFT can be integrated with model space discriminative training efficiently. We also compare GDFT with the state of the art feature space discriminative training algorithms and study different training procedures. Based on the proposed optimization algorithms, we discuss how to perform single pass discriminative training in chapter 6, Chapter 7 is about the future work and the conclusions.

Chapter 2

Background

2.1 Discriminative Objective Functions for Speech Recognition

The acoustic model, HMM, is often optimized for some objective functions during training. As a generative model, maximizing the likelihood on the train set is the standard approach and when Gaussian mixture is used as state emission probability, the likelihood can be expressed as,

$$\begin{aligned} F_{ML}(\theta) &= \sum_i \log P(X^{(i)}|W_i;\theta) \\ &= \sum_i \sum_t \sum_j -\frac{1}{2}\gamma_t(j)\{D\log(2\pi) + \log|\Sigma_j| + (x_t^{(i)} - \mu_j)'\Sigma_j^{-1}(x_t^{(i)} - \mu_j)\} \end{aligned}$$

where W_i is the reference word sequence of the i-th utterance in the train set with T frames; θ represents the HMM parameters which includes the mean vectors (μ_j) and the covariance matrices (Σ_j); $X^{(i)} \equiv \{x_1^{(i)}, \ldots, x_T^{(i)}\}$ is the observation of i-th utterance and each feature $x_t^{(i)}$ is a D-dimensional feature vector; $\gamma_t(j)$ is the posterior probability of choosing j-th Gaussian distribution at time t.

Maximizing the likelihood function, F_{ML}, can be done by the Baum-Welch (BW) al-

gorithm [Baum et al. (1970), Welch (2003)] which utilizes the Expectation-Maximization (EM) algorithm. More details will be given in section 2.2.1. Since F_{ML} contains some terms which are not related to the optimization of θ, an auxiliary function, Q is used instead to represent the likelihood during training,

$$Q(\theta) = \sum_t \sum_j \gamma_t(j)\{\log|\Sigma_j| + (x_t - \mu_j)'\Sigma_j^{-1}(x_t - \mu_j)\} . \quad (2.1)$$

We removed the utterance index i just for simplicity. This auxiliary function can be considered as a negative log likelihood function and minimizing Q on θ is equivalent to maximizing F_{ML}.

2.1.1 Maximum Mutual Information (MMI)

As mentioned, HMM is not a correct model for the human speech data, so ML estimation is not optimal in terms of classification or recognition accuracy. To optimize for the recognition performance, one can optimizes the posterior probability, $P(W|X;\theta)$, since the Bayes decision rule states that the classifier would achieve the minimum error if it makes decision based on the posterior probability. By the Bayes rule, the posterior can be decomposed into:

$$\begin{aligned} P(W|X) &= \frac{P(X|W)P(W)}{P(X)} \\ &= \frac{P(X|W)P(W)}{\sum_{W'} P(X|W')P(W')} \end{aligned} \quad (2.2)$$

where $P(X|W)$ is the likelihood and it is the HMM for speech recognition; $P(W)$ is the prior and it is the language model (LM) for speech recognition.

During recognition, θ is fixed so $P(X)$ is a constant. Hence, it can be ignored and the recognizer would search for a hypothesis W such that $P(W|X) \propto P(X|W)P(W)$ is maximized. However, in training, θ is not fixed, so $P(X)$ should also be considered during optimization.

Optimizing θ for the posterior probability is also known as maximum mutual informa-

tion estimation because the empirical mutual information is expressed as,

$$\begin{aligned} I(W,X) &= \frac{P(X|W)P(W)}{P(X)P(W)} \\ &= P(W|X) \times \frac{1}{P(W)} \, . \end{aligned} \tag{2.3}$$

Given $P(W)$ is uniformly distributed, $\frac{1}{P(W)}$ is a constant, so maximizing the posterior probability is equivalent to maximizing the mutual information.

Although the denominator of equation 2.2 considers all possible word sequences, in practice, it is approximated by a N-best list or more often, a lattice. When using lattice to represent the competing hypotheses, a path which represents the reference is often added to the lattice if the reference path is missing from the lattice. One can consider the lattice as a HMM with a directed acyclic topology. In such a case, the objective function can be simplified as

$$F_{MMI}(\theta) = \log P_r(X;\theta) - \log P_c(X;\theta) \, . \tag{2.4}$$

where P_r is the likelihood of the reference; P_c is the likelihood of the lattice with reference path attached. The prior probabilities like $P(W)$ and $P(W')$ are removed since we assume they are uniformly distributed. However, discriminative training in practice often uses an unigram language model instead of an uniform one. Hence, the likelihood should be adjusted according to the priors and it should be taken care by the forward algorithm when computing $\log P_r$ and $\log P_c$. In sum, maximizing F_{MMI} is equivalent to maximizing the posterior probability and the mutual information.

It is important to note that the first term of F_{MMI} is the same as F_{ML} which is the likelihood of the reference. The second term of F_{MMI} represents the competing hypotheses. As a result, MMI is computationally more expensive than ML approach since it needs to first generate a set of competing hypotheses, and second, the objective function involves more terms comparing to the ML objective function.

2.1.2 Minimum Phone Error (MPE)

As a discriminative objective function, MMI considers all competing hypotheses W' equal and aims to improve the overall performance on the train set. However, speech recognition is often evaluated by word error rate (WER) or phone error rate (PER), which considers every token in the hypotheses. As a result, one may argue competing hypotheses should not be considered equal, but one should look at the error rate of each individual competitor. This brings the interest to derive a discriminative objective function which can evaluate the error at a finer degree, and the resulting minimum phone error (MPE) [Povey (2003)] is one of the most popular discriminative objective functions to date, which optimizes the phone error.

The objective function of MPE is defined as,

$$F_{MPE}(\theta) = \sum_i \frac{\sum_{W_i'} P(X|W_i';\theta)P(W_i')A(W_i',W_i)}{\sum_{W_i'} P(X|W_i';\theta)P(W_i')} , \tag{2.5}$$

where $A(W_i', W_i)$ computes a raw phone accuracy for the competing hypothesis W_i' on the reference W_i which is the i-th utterance in the train set. Compared to MMI which numerator is the reference, the numerator of MPE consists of all possible word sequences which are weighed by the phone accuracy. The MPE objective function can be further rewritten as,

$$F'_{MPE}(\theta) = \sum_i \sum_{W_i'} P(W_i'|X;\theta)A(W_i',W_i) , \tag{2.6}$$

which $P(W_i'|X;\theta)$ is the model-based posterior probability of the word sequence W_i'.

The MPE objective function is very flexible in the sense that we can use word error instead of phone error. By doing so, it is known as minimum word error (MWE). However, previous research found that MPE can often outperform MWE [Povey (2003)].

2.1.3 Minimum Classification Error (MCE)

MCE is originally proposed for multiple category classification problem where it optimizes a smoothed error rate based on isolated tokens [Juang and Katagiri (1992b)]. Later,

it was generalized to optimize the string level error for speech recognition [Juang and Katagiri (1992a)]. Similar to MMI, the MCE objective function is based on the reference and the competing hypotheses. MCE defines a distance measure,

$$d(X,\theta) \;=\; \log \frac{P(X|W;\theta)}{[\frac{1}{N}\sum_{i=1,W_i\neq W}^{N} P^{\eta}(X|W_i;\theta)]^{\frac{1}{\eta}}} \tag{2.7}$$

where N is the number of competitors. To simulate the decision rule during decoding, $d(X,\theta) \geq 0$ implies incorrect classification. Based on this distance measure, the MCE objective function is defined as,

$$F_{MCE}(\theta) \;=\; \frac{1}{1+\exp^{-(ad(X,\theta)+b)}}\,, \tag{2.8}$$

which uses a sigmoid function to simulate the zero-one classification count. The advantage of using the sigmoid function is to filter the outliers. As the gradient of the sigmoid function approaches zero when $d \to -\infty$ or $d \to \infty$, the sigmoid function allows the optimization to focus on the instances that can be corrected instead of some very wrong or problematic utterances. The parameters a and b can be tuned to control the shape of the sigmoid function. Hence, it can control the rate of the optimization and region that the optimization should focus on. Optimization of MCE is based on gradient descent, or it is known as the generalized probabilistic descent for MCE. However, [He and Deng (2008)] showed that it is possible to restructure the MCE objective function so that it becomes a rational function that can be optimized by the extended Baum-Welch (EBW) algorithm.

2.1.4 Boosted Maximum Mutual Information (BMMI)

Proposed by [Povey et al. (2008)], BMMI is an extension to the MMI objective function. The BMMI objective function is defined as,

$$F_{BMMI}(\theta) \;=\; \log \frac{P(X|W;\theta)P(W)}{\sum_{W'} P(X|W';\theta)P(W')\exp(-b\times A(W',W))}\,. \tag{2.9}$$

where b is tunable parameter called boosting factor [Povey et al. (2008)]. Similar to the MPE objective function, BMMI uses an accuracy function A to evaluate the competing

hypotheses. However, this scaling is only applied on the denominator statistics. The implementation of BMMI is very simple: one may subtract the acoustic scores (i.e. log likelihood) in the lattice by $b \times A(W', W_i)$ during the forward backward pass to obtain the adjusted posterior probability, then the rest is the same as the MMI training. As shown in [Povey et al. (2008)], BMMI outperforms MMI and is as effective as MPE for both model space and feature space training. Both BMMI and MPE are considered to be state of the art in discriminative training.

2.2 Optimization Algorithms for Hidden Markov Model (HMM)

This section focuses on the optimization algorithms for HMM. We begin with the BW algorithm, which provides ML estimate, then the EBW algorithm which can optimize the HMM for different discriminative objective functions.

2.2.1 Baum-Welch Algorithm (BW)

HMM contains hidden state sequence which is not directly observable. Hence, it is not trivial to optimize the likelihood of a HMM with observable data, X. Consider,

$$\begin{aligned} \log P(X|\theta) &= \log(\sum_S P(X, S; \theta)) \\ &= \log(\sum_S P(X|S; \theta)P(S)), \end{aligned} \tag{2.10}$$

where S is the hidden state sequence of HMM. Since the complete log likelihood $P(X, S; \theta)$ is expressed as a summation within a log function. It is difficult to decouple the likelihood and the prior probabilities and this makes optimization difficult.

To handle this problem, the BW algorithm [Baum et al. (1970); Welch (2003)], which is based on the EM algorithm, does not optimize the log likelihood, $\log P(X|\theta)$, directly.

Instead, it optimizes the complete log likelihood, $P(X, S; \theta)$. By the rule of total probability,

$$\begin{aligned} \log P(X, S; \theta) &= \log P(S|X; \theta) + \log P(X; \theta) \\ \Rightarrow \log P(X; \theta) &= \log P(X, S; \theta) - \log P(S|X; \theta) \, . \end{aligned} \tag{2.11}$$

Then, assuming we have a new set of HMM parameters, θ' and we take expectation with respect to S, conditional on a given X and θ, we have

$$\underbrace{E_{S|X,\theta}[\log P(X; \theta')]}_{L(X, \theta')} = \underbrace{E_{S|X,\theta}[\log P(X, S; \theta')]}_{Q(\theta, \theta')} - \underbrace{E_{S|X,\theta}[\log P(S|X; \theta')]}_{H(\theta, \theta')} \tag{2.12}$$

which we have three components: $L(X, \theta')$, $Q(\theta, \theta')$ and $H(\theta, \theta')$.

The expected log likelihood term, $L(X, \theta')$, can be easily simplified,

$$\begin{aligned} E_{S|X,\theta}[\log P(X; \theta')] &= \sum_S \log P(X; \theta') P(S|X, \theta) \\ &= \log P(X; \theta') \sum_S P(S|X, \theta) \\ &= \log P(X; \theta') \end{aligned} \tag{2.13}$$

which is equivalent to the log likelihood of the observable data.

Consider the H function,

$$\begin{aligned} H(\theta, \theta') &= E_{S|X,\theta}[\log P(S|X; \theta')] \\ &= \sum_S \log P(S|X; \theta') P(S|X; \theta) \\ &\leq \sum_S \log P(S|X; \theta) P(S|X; \theta) \\ &= H(\theta, \theta) \, . \end{aligned} \tag{2.14}$$

$H(\theta, \theta') \leq H(\theta, \theta)$ is due to Jensen's inequality and it plays a key role in the BW and the EM algorithm.

The Q function is known as the auxiliary function. Given equation 2.13 and equation 2.14, consider,

$$\begin{aligned} L(X,\theta') - L(X,\theta) &= Q(\theta,\theta') - Q(\theta,\theta) - (H(\theta,\theta') - H(\theta,\theta)) \\ &= Q(\theta,\theta') - Q(\theta,\theta) - (\text{+ve}\Delta) \ . \end{aligned} \tag{2.15}$$

This implies $L(X,\theta') \geq L(X,\theta)$ if and only if $Q(\theta,\theta') \geq Q(\theta,\theta)$. Hence, one can optimize the auxiliary function Q instead of L.

Optimizing the Q function is easier since it no longer has the variable coupling issue as shown in equation 2.10. Consider,

$$\begin{aligned} \max_{\theta'} Q(\theta,\theta') &= E_{S|X,\theta}[\log P(X,S;\theta')] \\ &= \sum_{S} P(S|X;\theta) \log P(X,S;\theta') \\ &= \sum_{S} P(S|X;\theta)(\log P(X|S;\theta') + \log P(S;\theta')) \ . \end{aligned} \tag{2.16}$$

Given an initial model θ, we can search for θ' such that Q is maximized. We can obtain the BW update equations for continuous density HMM by taking the partial derivative of Q with respect to the means and covariances and set them to zero,

$$\mu_j^{BW} = \frac{\sum_t \gamma_t(j) x_t}{\sum_t \gamma_t(j)} \tag{2.17}$$

$$\Sigma_j^{BW} = \frac{\sum_t \gamma_t(j) x_t x_t'}{\sum_t \gamma_t(j)} - \mu_j \mu_j' \ . \tag{2.18}$$

2.2.2 Extended Baum-Welch Algorithm (EBW)

The EBW algorithm aims to derive an HMM update equation similar to the BW algorithm, but optimizing for some discriminative objective function. In the work conducted by [Gopalakrishnan et al. (1989, 1991)], an algorithm is developed to optimize rational objective functions for the discrete HMM. Since most discriminative objective functions are rational functions, the algorithm can perform discriminative training. The algorithm is based on the Baum-Eagon inequality for polynomials. The theory states that given a

polynomial, $f(x)$, with non-negative coefficients and real variables x_{ij} such that $x_{ij} \geq 0$ and $\sum_j x_{ij} = 1$, the transformation,

$$T(x_{ij}) \quad = \quad \frac{x_{ij}(\frac{\partial f}{\partial x_{ij}}(x))}{\sum_j x_{ij}(\frac{\partial f}{\partial x_{ij}}(x))}\,, \tag{2.19}$$

guarantees $f(T(x)) \geq f(x)$.

This theory is useful for discrete HMM optimization since it operates on a domain of discrete probability distributions ($x_{ij} \geq 0$ and $\sum_j x_{ij} = 1$) and the likelihood of discrete HMM can be easily expressed as such polynomial. However, discriminative objective functions are rational functions which are not the type of functions that the Baum-Eagon inequality is dealing with. To handle this problem, [Gopalakrishnan et al. (1989)] suggested given a rational function $r(x) = \frac{n(x)}{d(x)}$ such that $n(x)$ and $d(x)$ are polynomials fulfilling the constraints of Baum-Eagon inequality, one can optimize $g(x) = n(x) - r(x_0)d(x)$ instead of $r(x)$ directly (where x_0 represents some initial value of x). The reason is first $g(x_0) = 0$, hence, if there exists x such that $g(x) > g(x_0) = 0$, it implies,

$$\begin{aligned} & g(x) && > 0 \\ \Rightarrow \quad & n(x) - r(x_0)d(x) && > 0 \\ \Rightarrow \quad & \tfrac{n(x)}{d(x)} - r(x_0) && > 0 \\ \Rightarrow \quad & r(x) - r(x_0) && > 0\,. \end{aligned} \tag{2.20}$$

As a result, one can work on the polynomial $g(x)$ instead of the rational function $r(x)$. However, the coefficients of $g(x)$ may no longer be non-negative which is required by the Baum-Eagon inequality. Therefore, one can modify equation 2.19 to,

$$T_D(x_{ij}) \quad = \quad \frac{x_{ij}(\frac{\partial f}{\partial x_{ij}}(x) + D)}{\sum_j x_{ij}(\frac{\partial f}{\partial x_{ij}}(x) + D)}\,, \tag{2.21}$$

and it can be shown that as long as D is large enough, where D is a finite positive real number, the transformation, T_D, still guarantees $g(T_D(x)) \geq g(x)$ [Gopalakrishnan et al.

(1989)]. Finally, the update equations for discrete HMM which optimizes for any rational objective function, R, are,

$$\begin{aligned} \hat{a}_{ij} &= \frac{a_{ij}(\frac{\partial R}{\partial a_{ij}}(\theta) + D)}{\sum_j a_{ij}(\frac{\partial R}{\partial a_{ij}}(\theta) + D)} \\ \hat{b}_{ik} &= \frac{b_{ik}(\frac{\partial R}{\partial b_{ik}}(\theta) + D)}{\sum_k b_{ik}(\frac{\partial R}{\partial b_{ik}}(\theta) + D)} \end{aligned} \tag{2.22}$$

where a_{ij} is the transition probability from state i to state j; b_{ik} is the emission probability of output label k while being at state i; R is any rational objective function which can be any discriminative objective function in section 2.1; θ represents the current parameters of the discrete HMM.

Based on the work by [Gopalakrishnan et al. (1989, 1991)], the algorithm is extended for continuous density HMM with Gaussian mixtures [Normandin and Morgera (1991)]. The idea is to use infinitely many discrete distributions to approximate a Gaussian distribution. Figure 2.1 is an illustration to explain the approximation [Normandin and Morgera (1991)].

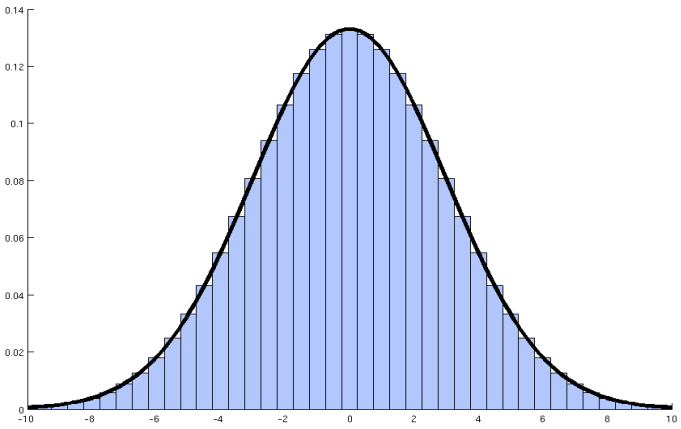

Figure 2.1: A figure showing the use of discrete distributions to approximate a Gaussian distribution.

Given the interval width, Δ, is small and restricting the discrete distributions to be

consistent to the Gaussian distribution they are approximating, the work in [Normandin and Morgera (1991)] derives an update equation for continuous density HMM and these equations are later known as the EBW update equations:

$$\mu_j^{EBW} = \frac{\sum_t \gamma_t^r(j)x_t - \sum_t \gamma_t^c(j)x_t + D_j\mu_j^0}{\sum_t \gamma_t^r(j) - \sum_t \gamma_t^c(j) + D_j}, \tag{2.23}$$

$$\Sigma_j^{EBW} = \frac{\sum_t \gamma_t^r(j)x_t x_t' - \sum_t \gamma_t^c(j)x_t x_t' + D_j(\Sigma_j^0 + \mu_j^0\mu_j^{0\prime})}{\sum_t \gamma_t^r(j) - \sum_t \gamma_t^c(j) + D_j} - \mu_j^{EBW}\mu_j^{EBW\prime} \tag{2.24}$$

where μ^0 and Σ^0 are the mean and covariance of the previous iteration and the superscript r and c denotes whether the posterior probability or the observation belongs to the numerator (reference) or the denominator (competitor) respectively.

Unlike the update equations of discrete HMM which the constant D is some finite number, in the formulation of [Normandin and Morgera (1991)], the value of D is related to the interval width Δ. When Δ tends to zero, D goes to infinity. Hence, the work in [Normandin and Morgera (1991)] concludes that these update equations are not guaranteed to converge. However, recent studies [He and Deng (2008)] found that it is possible to prove EBW's convergence given a finite D.

Since then, the EBW algorithm has shown to be effective in improving large scale speech recognition systems [Valtchev et al. (1997); Woodland and Povey (2000); Povey and Woodland (2001); Povey (2003)]. However, one remaining problem is how to set the D-term in the EBW update equations. While the convergence proofs from [Normandin and Morgera (1991)] and [He and Deng (2008)] do not give any guideline about tuning, this D value is often tuned empirically. One common heuristic proposed by [Povey and Woodland (2001)], is setting D_j to be the maximum of i) twice the value necessary to keep the covariance of the j-th Gaussian to be positive definite, or, ii) E times the denominator occupancy, where E is tuned empirically and its value is often between one and two.

Another technique which is also often applied to EBW is the I-smoothing [Povey and Woodland (2002); Povey (2003)]. I-smoothing can be considered as using a prior over the parameters of each Gaussian distribution. For the original I-smoothing, the prior is based

on ML statistics and the EBW update equations are extended as,

$$\hat{\mu}_j = \frac{\sum_t \gamma_t^r(j)x_t - \sum_t \gamma_t^c(j)x_t + D_j\mu_j^0 + \tau\mu_j^{ML}}{\sum_t \gamma_t^r(j) - \sum_t \gamma_t^c(j) + D_j + \tau} , \tag{2.25}$$

$$\hat{\Sigma}_j = \frac{\sum_t \gamma_t^r(j)x_t x_t' - \sum_t \gamma_t^c(j)x_t x_t' + D_j(\Sigma_j^0 + \mu_j^0\mu_j^{0\prime}) + \tau(\Sigma_j^{ML} + \mu_j^{ML}\mu_j^{ML\prime})}{\sum_t \gamma_t^r(j) - \sum_t \gamma_t^c(j) + D_j + \tau} - \hat{\mu}_j\hat{\mu}_j \tag{2.26}$$

where μ_j^{ML} and Σ_j^{ML} are ML estimates of the mean and covariance of j-th Gaussian using the statistics collected in the current iteration; τ is a tuning parameter for I-smoothing which needs to be tuned empirically. In most cases, τ is set to 100 for MMI-based objective functions or τ is set to 50 for MPE objective function. I-smoothing is not limited to using ML statistics. As shown in [Povey et al. (2008)], when MPE or BMMI is used, one can also use MMI statistics, or the statistics from the previous iteration to perform I-smoothing. It can further improve the performance of discriminative training.

2.2.3 Gradient Ascent and Its Relation to the EBW Algorithm

Before EBW was proposed, discriminative training was performed using gradient descent. Gradient ascent can be performed by computing the gradient of the objective function with respect to the model parameters to be optimized. For example, to optimize the acoustic model for the MMI objective function, one can compute the gradient for mean,

$$\begin{aligned}\frac{\partial F_{MMI}}{\partial \mu_j} &= \frac{\partial}{\partial \mu_j}(\log P_r(X;\mu_j) - \log P_c(X;\mu_j)) \\ &= \sum_t \gamma_t^r(j)\Sigma_j^{-1}(x_t - \mu_j) - \sum_t \gamma_t^c(j)\Sigma_j^{-1}(x_t - \mu_j) ,\end{aligned} \tag{2.27}$$

and the gradient for covariance,

$$\begin{aligned}\frac{\partial F_{MMI}}{\partial \Sigma_j} &= \frac{\partial}{\partial \Sigma_j}(\log P_r(X;\Sigma_j) - \log P_c(X;\Sigma_j)) \\ &= \frac{1}{2}\sum_t \gamma_t^r(j)[\Sigma_j^{-1} - \Sigma_j^{-1}(x_t - \mu_j)(x_t - \mu_j)'\Sigma_j^{-1}] \\ &- \frac{1}{2}\sum_t \gamma_t^c(j)[\Sigma_j^{-1} - \Sigma_j^{-1}(x_t - \mu_j)(x_t - \mu_j)'\Sigma_j^{-1}] ./\end{aligned} \tag{2.28}$$

Then, we obtain the update equations for gradient ascent [Schluter et al. (1997)],

$$\begin{aligned}
\tilde{\mu}_{jd} &:= \mu_{jd} + \frac{\lambda_\mu}{\sigma_{jd}^2}[\sum_t \gamma_t^r(j)x_t - \sum_t \gamma_t^c(j)x_t + \sum_t(\gamma_t^r(j) - \gamma_t^c(j))\mu_{jd}] \quad (2.29)\\
\tilde{\sigma}_{jd}^2 &:= \sigma_{jd} + \frac{\lambda_\sigma}{2\sigma_{jd}^4}[\sum_t \gamma_t^r(j)x_t^2 - \sum_t \gamma_t^c(j)x_t^2\\
&- 2(\sum_t \gamma_t^r(j)x_t - \sum_t \gamma_t^c(j)x_t)\mu_{jd}\\
&+ \sum_t(\gamma_t^r(j) - \gamma_t^c(j))(\mu_{jd}^2 - \sigma_{jd}^2)] \; . \quad (2.30)
\end{aligned}$$

where $\tilde{\mu}_{jd}$ and $\tilde{\sigma}_{jd}^2$ are the d-th dimension of the updated j-th Gaussian distribution respectively.

In [Schluter et al. (1997)], it was discovered that by choosing appropriate learning rates, λ_μ and λ_σ,

$$\begin{aligned}
\lambda_\mu &= \frac{\sigma_{jd}^2}{\sum_t \gamma_t^r(j) - \sum_t \gamma_t^c(j) + D_j} \quad (2.31)\\
\lambda_\sigma &= \frac{2\sigma_{jd}^4}{\sum_t \gamma_t^r(j) - \sum_t \gamma_t^c(j) + D_j} \; , \quad (2.32)
\end{aligned}$$

where D_j is the Gaussian specific constant used in the EBW algorithm. Then, the update equations for gradient ascent are very close to the EBW update equations,

$$\begin{aligned}
\tilde{\mu}_{jd} &= \frac{\sum_t \gamma_t^r(j)x_t - \sum_t \gamma_t^c(j)x_t + D_j\mu_{jd}}{\sum_t \gamma_t^r(j) - \sum_t \gamma_t^c(j) + D_j} \quad (2.33)\\
\tilde{\sigma}_{jd}^2 &= \frac{\sum_t \gamma_t^r(j)x_t^2 - \sum_t \gamma_t^c(j)x_t^2 + D_j(\sigma_{jd}^2 + \mu_{jd}^2)}{\sum_t \gamma_t^r(j) - \sum_t \gamma_t^c(j) + D_j} - \tilde{\mu}_{jd}^2 + (\mu_{jd} - \tilde{\mu}_{jd})^2 \quad (2.34)
\end{aligned}$$

where the mean update equation is the same as the EBW update equation and the covariance update equation has an extra term $(\mu_{jd} - \tilde{\mu}_{jd})^2$.

2.3 From Model Space to Feature Space Discriminative Training

Discriminative training is not only applicable to HMM optimization. Previous studies have shown that discriminative training can also optimize the features to improve recognition performance. Feature space discriminative training can be roughly divided into two types: one type focuses on the feature extraction process and another type is about feature transformation.

Discriminative feature extraction (DFE) optimizes components in the feature extraction process for some discriminative objective function like MCE. In [Biem and Katagiri (1993, 1994); Mak et al. (2002, 2003)], the filter-bank is optimized for MCE using gradient descent, and the optimized features was found to be effective in improving recognition performance of smaller tasks like TIDIGITS [Leonard (1984)] and Aurora [Pearce and Hirsch (2000)] corpora.

Another type of feature space discriminative training is often known as the discriminative feature transformation (DFT) methods. DFT leaves the feature extraction process the same but applies some discriminatively optimized transformation on the features. Popular DFT techniques include feature space MPE/MMI (fMPE/MMI) [Povey et al. (2005); Povey (2005); Povey et al. (2008)], and region dependent feature transformation (RDFT) [Zhang et al. (2006a,b)]. These techniques have proven to be effective on improving large scale speech recognition tasks.

2.3.1 Feature Space MPE/MMI (fMPE/MMI)

fMPE/MMI discriminative training algorithm performs linear transformation on the feature vectors, and the transformation is optimized for the MPE/MMI objective function. The basic form of fMPE/MMI [Povey et al. (2005)] is formulated as

$$z_t = x_t + M_1 h_t \ , \tag{2.35}$$

where x_t is the original feature at time t and we assume the feature space dimension is D; h_t is the Gaussian posterior vector computed by a Gaussian mixture model (GMM) on feature x_t; M_1 is a linear transform which is optimized for MMI/MPE using gradient ascent and z_t is the transformed feature vector. The GMM, either trained from the data or induced from the acoustic model, determines which transforms should be applied to the feature vectors.

The paper by [Zhang et al. (2006a)] shows that equation 2.35 can be rewritten as,

$$y_t = \sum_i \gamma_t(i)(x_t + b_i) = x_t + \sum_i \gamma_t(i) b_i \tag{2.36}$$

where b_i is a D-dimensional bias corresponding to the i-th row of M_1; $\gamma_t(i)$ is the posterior probability of Gaussian i at time t. From this point of view, we can consider the transformation of fMMI/MPE consists of a set of biases to be added to the features, and the weights, $\gamma_t(i)$, is determined by the GMM during recognition.

While this basic form of fMPE/MMI can improve the recognition performance, the feature transform in fMPE/MMI is extended to incorporate more features. One proposed extension is mean offset features [Povey (2005)]. It is done by expanding the vector h_t so that it does not only contain the posterior probabilities but also the mean-offset features, i.e. h_t is now redefined as:

$$\begin{aligned} h_t \equiv\ & [5\gamma_1, \gamma_1(x_t(1) - \mu_1(1))/\sigma_1(1), \gamma_1(x_t(2) - \mu_1(2))/\sigma_1(2), \ldots, \\ & 5\gamma_2, \gamma_2(x_t(1) - \mu_2(1))/\sigma_2(1), \gamma_2(x_t(2) - \mu_2(2))/\sigma_2(2), \ldots, \\ & 5\gamma_N, \gamma_N(x_t(1) - \mu_N(1))/\sigma_N(1), \gamma_N(x_t(2) - \mu_N(2))/\sigma_N(2), \ldots]' \,. \end{aligned}$$

Although the number of parameters increases as the dimension of h_t increases (h_t is now $N(D+1)$-dimension where N is the number of mixtures in the GMM), it was found fMPE/MMI with this extension requires fewer mixtures in the GMM (which leads to fewer biases). While the original fMPE/MMI needs hundreds of thousands mixtures, the new fMPE/MMI only needs a few thousands mixtures [Povey (2005)]. Therefore, the number of parameters remains tractable for training.

Another extension to fMMI/MPE is context training. In addition to the main transform, M_1, there is another layer of transformation for the features. After applying the main

transform, M_1, to the mean offset features, h_t, we obtain the offset vectors,

$$y_t = M_1 h_t \,. \tag{2.37}$$

Then, we apply the context transform, M_2,

$$z_{td} = x_{td} + \sum_{f=-F}^{F} M_{2,(f,d)} y_{t+f,d} \tag{2.38}$$

where z_{td} is the d-th dimension of the final feature vector z_t; $M_{2,(f,d)}$ represents the f-th row and d column of M_2.

The context transform, M_2 is optimized for the MMI/MPE objective function like the main transform. It is important to note that since both M_1 and M_2 are optimized using gradient ascent. They cannot be both zero matrices initially since otherwise, the gradients of M_1 and M_2 are always zero. To handle this situation, M_1 is initialized as a zero matrix, while M_2 is initialized in a way that the frames closer to the center frame get more weights compared to the frames further away. For example, considering a simple case that the context window has a size of ± 1 frame, M_2 can be initialized as

$$M_2 = \begin{bmatrix} \frac{1}{2}, \ldots, \frac{1}{2} \\ 1, \ldots, 1 \\ \underbrace{\frac{1}{2}, \ldots, \frac{1}{2}}_{\text{size of D}} \end{bmatrix} \tag{2.39}$$

which consists of 3 rows and D columns. This transform performs context expansion and produces the final feature vectors z_t. In general, the context matrix, M_2 should have $2F+1$ rows and D columns.

The last extension to fMMI/MPE described in [Povey (2005)] is block transformation. Effectively, it is having multiple context and main transforms. Suppose we have K blocks of transforms, it means,

$$y_t^k = M_1^k h_t \tag{2.40}$$

$$z_{td} = x_{td} + \sum_{k=1}^{K} \sum_{f=-F}^{F} \frac{1}{K} M_{2,(f,d)}^k y_{t+f,d}^k \,. \tag{2.41}$$

In addition, the factor $\frac{1}{K}$ can be merged into M_2^k and optimized together, Hence, we have the final fMMI/MPE equation for computing the features,

$$z_{td} \quad = \quad x_{td} + \sum_{k=1}^{K} \sum_{f=-F}^{F} M_{2,(f,d)}^{k} y_{t+f,d}^{k} \; . \tag{2.42}$$

Although the sum of linear transforms is equivalent to using one single linear transform, in practice, using multiple transforms improves the performance of fMPE/MMI. The initialization remains the same for M_1^k and M_2^k except all entries in M_2^k needs to add a small random numbers to make sure that the transforms will not go to the same direction during the gradient ascent.

The training procedure of fMPE/MMI is iterative and each iteration involves three passes on the data [Povey (2005)]. The first pass collects the MPE/MMI statistics. This statistics are called indirect statistics which are required to augment the gradients. More details on this subject will be discussed in section 2.3.3. The second pass collects the fMPE/MMI statistics for the purpose of performing gradient ascent on the main and the context transforms. The final pass is performing ML update for the HMM using the new features from the second pass. The whole process is repeated four times. After the features are optimized for MPE/MMI objective function, one can further improve recognition performance by using the model space MPE/MMI.

2.3.2 Region Dependent Feature Transformation (RDFT)

Region dependent feature transformation (RDFT) [Zhang et al. (2006a,b)] is similar to fMPE/MMI in the sense that it uses a GMM to divide the feature space into different regions, and each region has its own specific transform. The final feature of RDFT is defined as a weighted average of all region specific features,

$$z_t = \sum_i \gamma_t(i) f_i(x_t) \; . \tag{2.43}$$

When the transformation is linear, this form of RDFT is called region dependent linear

transformation (RDLT). The final features are computed by,

$$z_t = \sum_i \gamma_t(i)(A_i x_t + b_i) \ , \tag{2.44}$$

where A_i is the transformation matrix optimized for MPE/MMI objective. Compared to equation 2.36 of fMPE/MMI, RDLT is a more general form of fMPE/MMI, since it consists of the transformation matrices, A_i, in addition to the biases. Similar to fMPE/MMI, RDLT can also perform context training. In such a case, A_i becomes a projection matrix to project the concatenated feature supervectors back to the feature space. The supervectors may also contain the posterior features like fMPE/MMI if needed. For optimization, RDLT uses a quasi-Newton algorithm which uses gradient information to approximate the Hessian matrix for performing update like Newton method.

Zhang et al. (2006b) show fMPE/MMI with mean offset features can be rewritten as a form of RDLT. To simplify the discussion, the authors assume no context training. Then, by defining,

$$\gamma_t = [\gamma_t(1), \ldots, \gamma_t(N)]' \tag{2.45}$$

$$\delta_t = [\gamma_t(1)(x_t - \mu_1)'\Sigma_1^{-1'}, \ldots, \gamma_t(N)(x_t - \mu_N)'\Sigma_N^{-1'}]' \ , \tag{2.46}$$

one can reorganize the equation 2.35 into,

$$z_t = x_t + M_a \delta_t + M_b \gamma_t \ , \tag{2.47}$$

which breaks the transform M into two parts: M_a and M_b. M_a is a transform applied to the mean-offset features and its dimension is $d \times Nd$; M_b is a transform applied to the posterior features and its dimension is $d \times N$. Since $\sum_{i=1}^{N} \gamma_t(i) = 1$, we can further rewrite equation 2.47 as,

$$\begin{aligned} z_t &= x_t + \sum_{i=1}^{N} (\gamma_t(i) M_a^{(i)} \Sigma_i^{-1}(x_t - \mu_i) + \gamma_t(i) M_b^{(i)}) \\ &= \sum_{i=1}^{N} \gamma_t(i)[(I + M_a^{(i)}\Sigma_i^{-1})x_t + (M_b^{(i)} - M_a^{(i)}\Sigma_i^{-1}\mu_i)] \ . \end{aligned} \tag{2.48}$$

where $M_a^{(i)}$ is the i-th $d \times d$ block of M_a and $M_b^{(i)}$ is the $i - th$ column of M_b.

Zhang et al. (2006b) concludes that this can be considered as a constrained version of RDLT which restricts the transform in equation 2.44 to be,

$$A_i = I + M_a^{(i)}\Sigma_i^{-1} \tag{2.49}$$
$$b_i = M_b^{(i)} - M_a^{(i)}\Sigma_i^{-1}\mu_i \; . \tag{2.50}$$

In sum, RDLT can be considered as a more general form of fMPE/MMI.

2.3.3 Optimization and Indirect Statistics for fMPE/MMI and RDLT

Optimization for fMPE/MMI and RDLT is performed by gradient ascent or quasi-Newton method. However, instead of using the gradient to update the feature transforms, all gradients are augmented by a term called indirect statistics [Povey et al. (2005)]. In this section, we describe what this means from the optimization perspective.

To simplify the discussion, we assume that we use the gradient descent for optimization, and MMI for the discriminative objective function. Let Q_{MMI} be the auxiliary function which represents the negative of the MMI objective function, i.e.

$$\begin{aligned} Q_{\mathsf{MMI}} &= \sum_t \sum_m \gamma_t^r(m)[\log|\Sigma_m| + (z_t - \mu_m)'\Sigma_m^{-1}(z_t - \mu_m)] \\ &- \sum_t \sum_m \gamma_t^c(m)[\log|\Sigma_m| + (z_t - \mu_m)'\Sigma_m^{-1}(z_t - \mu_m)] \; . \end{aligned} \tag{2.51}$$

Minimizing Q_{MMI} is equivalent to maximizing the mutual information, and z_t is the final feature vector which is also a function of the feature transforms that we are optimizing. Also, we assume that we use a context window of size $\pm F$ frames and K blocks of transforms. To perform gradient descent, we need to compute the gradient with respect to the

main transform, M_1^k and the context transform, M_2^k,

$$\left(\frac{\partial Q_{\text{MMI}}^{\text{direct}}}{\partial M_1^k}\right)_{(i,j)} = \sum_t \left(\frac{\partial Q_{\text{MMI}}^{\text{direct}}}{\partial z_t}\right)_i \sum_{f=-F}^{F} M_{2,(f+F,i)}^k h_{t+f,j} \quad (2.52)$$

$$\left(\frac{\partial Q_{\text{MMI}}^{\text{direct}}}{\partial M_2^k}\right)_{(f+F,i)} = \sum_t \left(\frac{\partial Q_{\text{MMI}}^{\text{direct}}}{\partial z_t}\right)_i y_{t+f,i}^k \quad (2.53)$$

$$\frac{\partial Q_{\text{MMI}}^{\text{direct}}}{\partial z_t} = \sum_m \gamma_t^r(m)\Sigma_m^{-1}(z_t - \mu_m) - \sum_m \gamma_t^c(m)\Sigma_j^{-1}(z_t - \mu_m) \ , \quad (2.54)$$

where $(\frac{\partial Q_{\text{MMI}}^{\text{direct}}}{\partial z_t})_i$ represents the i-th element of in the vector $\frac{\partial Q_{\text{MMI}}^{\text{direct}}}{\partial z_t}$ and the index (i,j) represents the row and the column of the corresponding matrices.

The gradients in equation 2.52 and 2.53 are called direct statistics. However, instead of using these gradients to update the feature transforms, the gradients are augmented by,

$$\frac{\partial Q_{\text{MMI}}}{\partial M_1^k} = \frac{\partial Q_{\text{MMI}}^{\text{direct}}}{\partial M_1^k} + \frac{\partial F_{\text{MMI}}^{\text{indirect}}}{\partial M_1^k} \quad (2.55)$$

$$\frac{\partial Q_{\text{MMI}}}{\partial M_2^k} = \frac{\partial Q_{\text{MMI}}^{\text{direct}}}{\partial M_2^k} + \frac{\partial F_{\text{MMI}}^{\text{indirect}}}{\partial M_2^k} \quad (2.56)$$

where

$$\left(\frac{\partial Q_{\text{MMI}}^{\text{indirect}}}{\partial M_1^k}\right)_{(i,j)} = \sum_t \left(\frac{\partial Q_{\text{MMI}}^{\text{indirect}}}{\partial z_t}\right)_i \sum_{f=-F}^{F} M_{2,(f+F,i)}^k h_{t+f,j} \quad (2.57)$$

$$\left(\frac{\partial Q_{\text{MMI}}^{\text{indirect}}}{\partial M_2^k}\right)_{(f+F,i)} = \sum_t \left(\frac{\partial Q_{\text{MMI}}^{\text{indirect}}}{\partial z_t}\right)_i y_{t+f,i}^k \quad (2.58)$$

$$\frac{\partial Q_{\text{MMI}}^{\text{indirect}}}{\partial z_t} = \sum_m \frac{\gamma_t^r(m)}{\sum_{t'}\gamma_{t'}^r(m)}\left(\frac{\partial Q_{\text{MMI}}}{\partial \mu_m} + 2\frac{\partial Q_{\text{MMI}}}{\partial \Sigma_m}(z_t - \mu_m)\right) . \quad (2.59)$$

Equation 2.57 and 2.58 are derived by assuming the means and the covariances in the acoustic model are functions of the feature transforms, and the functions are the BW update equations, i.e.

$$\mu_m = \frac{\sum_t \gamma^r(m) z_t}{\sum_t \gamma_t^r(m)} \quad (2.60)$$

$$\Sigma_m = \frac{\sum_t \gamma^r(m) z_t z_t'}{\sum_t \gamma_t^r(m)} - \mu_m \mu_m' \ . \quad (2.61)$$

To compute the indirect statistics in equation 2.57 and 2.58, we need to compute

$$\frac{\partial Q_{\text{MMI}}}{\partial \mu_m} = \kappa \Sigma_m^{-1} (\sum_t \gamma_t^r(m) z_t - \sum_t \gamma_t^r(m) z_t - \mu_m (\sum_t \gamma_t^r(m) - \gamma_t^c(m))) \quad (2.62)$$

$$\begin{aligned} \frac{\partial Q_{\text{MMI}}}{\partial \Sigma_m} &= \frac{\kappa}{2} (\sum_t \gamma_t^r(m)) (S_m^r \Sigma_m^{-1} \Sigma_m^{-1} - \Sigma_m^{-1}) \\ &- \frac{\kappa}{2} (\sum_t \gamma_t^c(m)) (S_m^c \Sigma_m^{-1} \Sigma_m^{-1} - \Sigma_m^{-1}) \end{aligned} \quad (2.63)$$

where κ is the acoustic model scale which is often the inverse of the grammar factor [Povey (2003)], and,

$$S_m^r = \frac{1}{\sum_t \gamma_t^r(m)} (\sum_t \gamma_t^r(m) z_t z_t' - 2 \sum_t \gamma_t^r(m) z_t \mu_m' + \sum_t \gamma_t^r(m) \mu_m \mu_m') \quad (2.64)$$

$$S_m^c = \frac{1}{\sum_t \gamma_t^c(m)} (\sum_t \gamma_t^c(m) z_t z_t' - 2 \sum_t \gamma_t^c(m) z_t \mu_m' + \sum_t \gamma_t^c(m) \mu_m \mu_m') \quad (2.65)$$

Note that some entries in equation 2.57 and 2.58, including S^r, S^c, $\sum_t \gamma_t^r(m)$, require the statistics of the whole train set. As a result, one has to perform a standard EBW pass to collect such statistics before computing the gradients for fMPE/MMI. A standard EBW pass is sufficient since it collects statistics like $\sum_t \gamma_t^{r/c}(m)$, $\sum_t \gamma_t^{r/c}(m) z_t$, $\sum_t \gamma_t^{r/c}(m) z_t z_t'$, which are sufficient to reconstruct the indirect statistics during the gradient computation.

It is important to note that the gradient used in fMPE/MMI consists of two components: direct statistics and indirect statistics. In which, the indirect statistics consists of the indirect gradient for means ($\frac{\partial Q_{\text{MMI}}}{\partial \mu_m}$), and the indirect gradient for covariances ($\frac{\partial Q_{\text{MMI}}}{\partial \Sigma_m}$). Since the gradients are summed together, it implies that the objective function of fMPE/MMI is a multi-objective function. Hence, when one claims fMMI is using MMI as the objective function, in fact, there are three MMI objective functions in the optimization problem: the standard MMI objective function, the MMI objective function which treats means as functions of the feature transforms, and another MMI objective function which treats covariances as functions of the feature transforms. With the same argument, since RDLT also uses indirect statistics when it computes the gradients, one can argue RDLT also optimizes a multi-objective function.

Once the gradient is computed, we can perform gradient descent using these update equations,

$$M^k_{1,(i,j)} := M^k_{1,(i,j)} + \nu^k_{1,(i,j)} \frac{\partial F_{\mathsf{MMI}}}{\partial M^k_{1,(i,j)}} \tag{2.66}$$

$$M^k_{2,(i,j)} := M^k_{2,(i,j)} + \nu^k_{2,(i,j)} \frac{\partial F_{\mathsf{MMI}}}{\partial M^k_{2,(i,j)}} \tag{2.67}$$

where

$$\nu^1_{(i,j)} = \frac{\sigma_i}{E_1(p^k_{1,ij} + n^k_{1,ij})} \tag{2.68}$$

$$\nu^2_{(i,j)} = \frac{1}{E_2(p^k_{2,ij} + n^k_{2,ij})} \,. \tag{2.69}$$

σ_i is the average standard deviation of the Gaussians in the current acoustic model in the i-th dimension; ν_1 and ν_2 are the learning rates for M_1 and M_2 respectively. p_{ij} and n_{ij} are computed by accumulating the positive parts and the negative parts of $\frac{\partial F}{\partial M}$ respectively.

The learning rates are controlled by the parameters E_1 and E_2 which need to be tuned. According to [Povey (2005)], E_1 and E_2 are adjusted so that no more than 10% of the paramters on the n-th iteration are on the opposite side of the value on the $n-2$-th iteration from the value on the $n-1$-th iteration. This heuristic is to prevent possible divergence during optimization. This is also known as the smooth update and it can only be applied starting from the second EM iteration.

Chapter 3

Baseline Automatic Speech Recognition Systems

We describe the baseline transcription systems used for evaluating different discriminative training algorithms in this chapter. These systems are built for large scale DARPA evaluations. The systems include an Iraqi Arabic ASR system, a Farsi ASR system and a modern standard Arabic (MSA) ASR system. The Iraqi and the Farsi systems are developed for the DARPA Spoken Language Communication and Translation System for Tactical Use program (TransTac) while the MSA ASR system is developed for the DARPA Global Autonomous Language Exploitation program (GALE).

The goal of the TransTac program is to develop effective, real-time, field portable, two-way speech-to-speech translation system for English and some low resource languages like Iraqi Arabic and Farsi. The program aims to develop a system which can facilitate communication between US military personnel and a foreign language speaker. Hence, the system is mainly designed for domains like force protection, medical screening and civil affairs. The translation system consists of three components: a speech recognition module, a machine translation module and a speech synthesis module. We develop the systems for both the PDA platform and laptop platform. However, for the experiments described in this thesis, we focus on the performance of the laptop system.

The goal of the GALE program is to develop and deploy the capability to absorb, analyze and interpret huge volumes of speech and text in modern standard Arabic and Mandarin Chinese, and make them available in English. The domain includes broadcast news and broadcast conversations from various radio and TV channels. In this thesis, we focus on the performance of the MSA ASR system.

3.1 Iraqi ASR System

Iraqi Arabic is the spoken form of Arabic used by the people of Iraq in everyday conversations. It is different from the MSA used in written communication. Since Iraqi Arabic is normally not written, a transcription convention is defined under the TransTac program for the purpose of data collection. Throughout the program, over 500 hours of Iraqi speech data are collected and transcribed which consists of over four million words in the transcription.

Our Iraqi ASR system is a single pass, speaker adaptive system which runs at real-time. Since it is a single pass decoding process, speaker adaptation is performed incrementally, which uses the previous hypotheses for unsupervised adaptation. Speaker statistics can be reset if it is switched to another user. Speaker adaptation is performed using constrained maximum likelihood linear regression (CMLLR) [Digalakis et al. (1995)] and maximum likelihood linear regression (MLLR) [Leggetter and Woodland (1995)] to adapt the acoustic model. To speed up the recognizer, Gaussian selection is used to speed up the decoding process to make sure the ASR system runs at real-time.

The acoustic model of the Iraqi ASR system is a 3-state sub-phonetically tied semi-continuous HMM-based recognizer composed of 7000 context dependent triphone models. Each model consists of a mixture of 64 Gaussians at the most, where the exact number of mixtures is determined by a merge-and-split training algorithm. Semi-tied covariance [Gales (1999)] and speaker adaptive training are also performed. This acoustic model is trained with 450 hours of Iraqi Arabic 16kHz speech data including data sets from Appen/BBN, Cepstral, IBM/DLI Pendleton, and Marine Acoustics Inc. The speech data is

represented by the first 13 Mel Frequency Cepstral Coefficients (MFCC) and power with a 10ms frame-shift and a 20ms Hamming window, together with approximations of the first and second derivatives. Frames with a context window of size ± 7 are concatenated to form supervectors and linear discriminant analysis (LDA) is applied to project the supervectors back to the dimensionality of 42 coefficients. In sum, this is the acoustic model for the ML system used in the experiments.

Under the TransTac program, a pronunication dictionary is provided by LDC. However, the dictionary does not cover all the words appeared in the train set. A standard CART-based technique is applied [Black et al. (1998)], which is an automatic grapheme/phoneme alignment technique, to find initial alignment. Hence, the letter to sound rules could be built without any knowledge of the target language, and construct the pronunication dictionary used for building the ASR system.

The language model (LM) for Iraqi ASR is a trigram model using modified Kneser-Ney smoothing. The training set consists of 4.5M words including data from the transcription of the audio training data. The system selects 62k vocabulary as its search vocabulary and it is based on frequency count.

To evaluate the performance of this Iraqi ASR system, we use the TransTac Iraqi Jun08 offline open set as the development set and the TransTac Iraqi Nov08 offline open set as the unseen test set. Each of these test set consists of one hour of Iraqi audio data in total. The test sets include in domain conversations between English speakers and Iraqi Arabic speakers. We only evaluate the performance on the Iraqi portion.

3.2 Farsi ASR System

The Farsi acoustic model has the very same topology as the Iraqi ASR system. It is trained with about 110 hours of Farsi 16kHz speech data collected by Appen, DLI, and University of Southern California. The acoustic model consists of 3000 models, each has at most 64 Gaussians which is determined by merge-and-split training. The acoustic model is bootstrapped from the Iraqi acoustic model. The two phones of Farsi not covered by the

Iraqi phone set are initialized by phones of the same phone category. After this phone mapping a first Farsi context independent acoustic model is bootstrapped from the Iraqi acoustic model. This first Farsi context independent system is used to force-align all data. Based on these new forced alignments we initialize a second context independent system before using our regular context dependent training routine. The pronunication dictionary is constructed in the same way for the Iraqi ASR system and the feature extraction process remains the same as the Iraqi ASR system as well.

The language model is a trigram model using modified Kneser-Ney smoothing, and it is trained with 900K words. The vocabulary size is around 33K words, which consists of all available words in the provided transcription under the DARPA TransTac program.

To evaluate the performance of this Farsi ASR system, we use the TransTac Farsi Jul07 offline open set as the test set. This test set consists of one hour of Farsi audio data in total.

3.3 Modern Standard Arabic ASR System

The Modern Standard Arabic (MSA) [Noamany et al. (2007)] ASR system is developed for the DARPA GALE Speech-to-Text evaluation. Unlike the Iraqi ASR system, this system is optimized for recognition performance without the real-time constraint.

This Arabic system is trained on approximately 1150 hours of training data, taken from the GALE P2 and P3 sets using both a vowelized, and an unvowelized dictionary. The training data provides manual segmentation and speaker clusters, while for the testing data, clusters have been generated automatically.

For feature extraction, we compute power spectral features using a FFT with a 10ms frame-shift and a 16ms Hamming window from the 16kHz audio signal. We use 13 MFCC per frame and perform cepstral mean and variance normalization on a cluster basis, followed by vocal tract length normalization (VTLN) [Lee and Rose (1996)]. To incorporate dynamic features, we concatenate 15 adjacent MFCC frames (± 7) and project 195 dimensional features into a 42 dimensional space using LDA transform. After LDA, we apply semi-tied covariance and speaker adaptive training.

For the development of our GMM based context dependent acoustic models, we apply an entropy-based polyphone decision tree to cluster the quinphones with context width ± 2. The system uses 6000 phonetically tied quinphones with at most 150 Gaussians per state, assigned using merge and split training, with diagonal covariance matrices.

During decoding, automatic speaker clustering of manually segmented audio is performed. Segments are clustered into speaker-specific clusters using Bayesian Information Criterion (BIC) to enable cluster-specific adaptation and normalization [Jin and Schultz (2004)].

The language model is trained from a variety of sources. The Arabic Gigaword corpus distributed by LDC is the major text resource for language modeling. In addition, we harvested transcripts from Al-Jazeera, Al-Akhbar, and Akhbar Elyom, as described in [Noamany et al. (2007)]. Acoustic transcripts from FBIS, TDT-4, GALE broadcast news (BN) and broadcast conversations (BC) are also used. The total number of words in the corpus amounted to 1.1 billion. The final LM is a 4-gram LM with 692M n-grams and 737K words in vocabulary.

Arabic is a phonetic language with a close correspondence between its letters and sounds. The pronunciation dictionary is generated using grapheme-to-phoneme rules. The unvowelized system contains 37 phones with 3 special phones for silence, non-speech events and non-verbal effects such as hesitations. We preprocess the text by mapping the 3 shapes of the grapheme for glottal stops to one shape at the beginning of the word since these are frequently miss-transcribed. This approach gives improvements in perplexity and final WER in our previous experiments.

The decoding process has three passes: 1) unvowelized speaker independent (UnvowSI) decoding, 2) unvowelized speaker adaptive (UnvowSA) decoding using the Unvow SI hypotheses for adaptation, and 3) vowelized speaker adaptive (VowSA) decoding using the Unvow SA hypotheses for adaptation. In addition, a small unvowelized speaker adaptive system using only 50 hours of training data is built (UnvowSA 50-hr). The purpose of building this system is to quickly test the performance of different configurations of some discriminative training algorithms. This system uses the input from the UnvowSI system to perform speaker adaptation. Figure 3.1 illustrates the multi-pass ASR system for MSA.

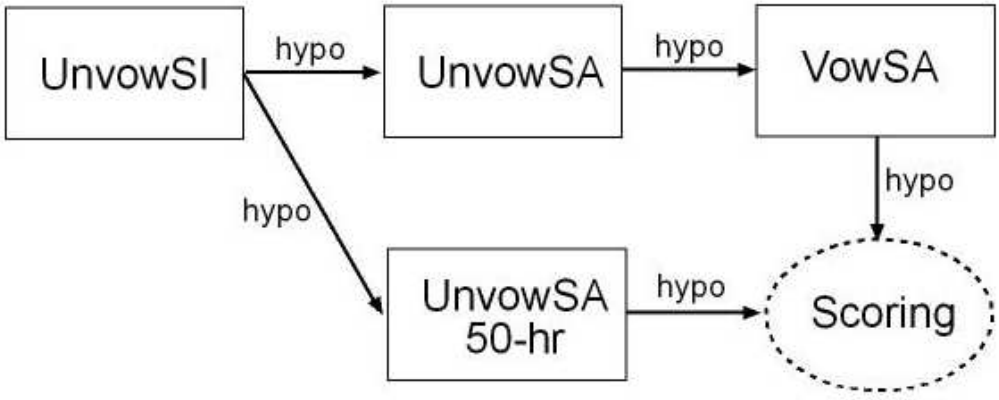

Figure 3.1: The overview of the MSA ASR system.

The MSA ASR system is evaluated using the GALE development and evaluation test sets. In this thesis, dev07, dev08 and dev09 are used as the development sets and eval09 and a subset of dev10 are used as the unseen test sets. All these test sets consist of mixtures of BN and BC data collected from various sources as shown in table 3.1. The development sets including dev07, dev08, and dev09 contain roughly two to three hours of audio data while eval09 has around five to six hours of data. The test set, dev10, has over five hours of data but a three hours subset is selected for evaluation.

Table 3.2 summarizes the ASR systems described in this chapter. We evaluate the performance of different discriminative training algorithms on these systems in later chapters.

Test sets	Sources
dev07	ABUDHABI ALAM ALJZ ARABIYA DUBAISCO IRAQIYAH KUWAITTV LBC SCOLA SYRIANTV
dev08	ALAM ALHIWAR ALHURRA ALJZ ALMANAR ALURDUNYA ARABIYA DUBAISCO IRAQIYAH KUWAITTV LBC OMANTV SAUDITV SCOLA SYRIANTV
dev09	ABUDHABI ALAM ALBAGHDADIA ALFAYHA ALHIWAR ALHURRA ALJZ ALURDUNYA ARABIYA DUBAI IRAQIYAH LBC OMANTV SAUDITV SCOLA YEMENTV
eval09	ABUDHABI ALAM ALBAGHDADYA ALFAYHA ALHIWAR ALHURRA ALJZ ALSHARQIYA ALURDUNYA ARABIYA DUBAI IRAQIYAH LBC SAUDITV SAWA SCOLA SYRIANTV YEMENTV
dev10	ABUDHABI ALAM ALBAGHDADYA ALHIWAR ALHURRA ALJZ ARABIYA IRAQIYAH LBC SCOLA SYRIANTV SAWA YEMENTV

Table 3.1: Sources of the GALE development and evaluation test sets.

	Iraqi ASR	Farsi ASR	Vow MSA ASR	Unvow MSA ASR
Train data	450 hr	110 hr	1100 hr	50 hr
System type	SA, 1-pass	SI, 1-pass	SA, 3-pass	SA, 2-pass
Vocab size	62k	33k	737k	737k
Adaptation	Incremental	None	Batch	Batch
# Gaussians	308k	112k	867k	52k
LM	3-gram	3-gram	4-gram	4-gram

Table 3.2: Description of the Iraqi, Farsi and MSA ASR systems.

Chapter 4

Generalized Baum-Welch Algorithm (GBW)

4.1 Introduction

We describe the Generalized Baum Welch (GBW) algorithm in this chapter which was first introduced in [Hsiao et al. (2009)]. In chapter 1 and 2, we discuss the optimization problem of discriminative training. The difficulty comes from the complicated non-convex objective function and the unbounded issue. In this chapter, we propose the GBW algorithm and we show that by transforming the optimization problem, we can handle these issues. The formulation of GBW shows both BW and EBW algorithms are special cases of GBW, and it also reveals an interesting connection between information theory and the EBW algorithm. Through the GBW algorithm, it helps us to understand the heuristics used in the EBW algorithm, and based on these insights, we can develop better variants of the EBW algorithm.

4.2 Bounding the Solution by Limiting Likelihood Changes

Following the discussion in section 1.1, optimizing the log likelihood difference function F in equation 1.1 can be unbounded to some parameters. However, we can address this unbounded issue easily by transforming the objective function in equation 1.1 into,

$$G(X,\theta) = |Q_r(X,\theta) - C_r| + |Q_c(X,\theta) - C_c| \tag{4.1}$$

where Q_r and Q_c are the negative log likelihood of the reference and the competing hypothesis respectively (see equation 2.1); C_r and C_c are the chosen target values that we want Q_r and Q_c to achieve respectively. The competing hypothesis is often represented by a lattice. Although we call the lattice as the competitor, the lattice is often attached with a path which represents the reference. This is a practice that is known to improve the performance of discriminative training as shown in [Valtchev et al. (1997)]. However, for simplicity, we just call Q_c as the competitor.

For this particular example, we choose the target values such that $Q_r(X,\theta) > C_r$ and $Q_c(X,\theta) < C_c$. As a result, by minimizing the function G, we are maximizing the log likelihood difference between the reference and the competitor, but we only want it to achieve the target values that we have chosen. In general, we have multiple files and each file has possibly multiple competitors. Hence, the formulation can be generalized as,

$$G(X,\theta) = \sum_i |Q_i(X,\theta) - C_i| \ . \tag{4.2}$$

Note that this formulation is very flexible that we can represent references and competitors at different granularity levels, since Q can be a likelihood function at the utterance or lattice level, or it can be a likelihood function for a word arc or a phone arc in the lattice. Generally speaking, we can have multiple terms for reference and competitors and each term has its own target value, C_i. It is also important to note that when each term corresponds to a word arc or a phone arc, not every term has equal importance because of different posterior probability in the lattice. To reflect this, one may add a weighting factor for each term or scale the target values. The formulas shown here, however, assume each term represents either a whole utterance (reference) or a lattice (competitor) for simplicity.

In addition, we can add a regularization function in order to control the optimization. Let $R(\theta, \theta^0)$ be a regularization function with θ^0 as the backoff model. Then, the objective function becomes,

$$G(X, \theta) = \sum_i |Q_i(X, \theta) - C_i| + R(\theta, \theta^0) \tag{4.3}$$

Although the function G in equation 4.3 remains to be non-convex, this formulation has an obvious advantage over the original problem – the unbounded issue no longer exists since G must be larger than or equal to zero. One easy way to define the target values is to encourage higher likelihood for the reference and lower likelihood for the competing hypotheses. This scheme is equivalent to MMI estimation.

4.3 Lagrange Relaxation

To minimize the function G, we may first transform the problem to,

$$\begin{array}{ll} \min\limits_{\epsilon,\theta} & \sum_i \epsilon_i + R(\theta, \theta^0) \\ \text{s.t.} & \epsilon_i \geq Q_i(\theta) - C_i \quad \forall i \\ & \epsilon_i \geq C_i - Q_i(\theta) \quad \forall i \ , \end{array}$$

where ϵ_i is a slack variable for the i-th term in equation 4.3. This optimization problem is equivalent to the original unconstrained problem in equation 4.3. We call this the primal problem of the GBW algorithm.

For simplicity, we first show the formulation for optimizing the mean vectors, and this formulation uses Mahalanobis distance as the regularization function on the means. The primal problem becomes,

$$\begin{array}{lll} \min\limits_{\epsilon,\mu} & \sum_i \epsilon_i & + \sum\limits_j D_j ||\mu_j - \mu_j^0||^2_{\Sigma_j} \\ \text{s.t.} & \epsilon_i \geq Q_i(\mu) - C_i \quad \forall i & \\ & \epsilon_i \geq C_i - Q_i(\mu) \quad \forall i \ , & \end{array} \tag{4.4}$$

where D_j is a Gaussian specific constant to control the weight of the regularization term; μ_j^0 is the mean vector that we want GBW to backoff to, and it is assumed to be model from the previous EM iteration.

We can then construct the Lagrangian dual for the primal problem. The Lagrangian is defined as,

$$\begin{aligned} L_m^P(\epsilon, \mu, \alpha, \beta) &= \sum_i \epsilon_i - \sum_i \alpha_i(\epsilon_i - Q_i(\mu) + C_i) \\ &- \sum_i \beta_i(\epsilon_i - C_i + Q_i(\mu)) \\ &+ \sum_j D_j ||\mu_j - \mu_j^0||^2_{\Sigma_j} \end{aligned} \tag{4.5}$$

where $\{\alpha_i\}$ and $\{\beta_i\}$ are the Lagrange multipliers for the first and the second set of constraints of the primal problem in equation 4.4. The Lagrangian dual is then defined as,

$$L_m^D(\alpha, \beta) = \inf_{\epsilon, \mu} L_m^P(\epsilon, \mu, \alpha, \beta) \tag{4.6}$$

Now, we can differentiate L_m^P w.r.t. μ and ϵ. Hence,

$$\frac{\partial L_m^P}{\partial \epsilon_i} = 1 - \alpha_i - \beta_i \tag{4.7}$$

$$\begin{aligned} \frac{\partial L_m^P}{\partial \mu_j} &= \sum_i (\alpha_i - \beta_i) \frac{\partial Q_i}{\partial \mu_j} + D_j \frac{\partial}{\partial \mu_j} ||\mu_j - \mu_j^0||^2_{\Sigma_j} \\ &= \sum_i (\alpha_i - \beta_i)(-2 \sum_t \gamma_t^i(j) \Sigma_j^{-1}(x_t - \mu_j)) \\ &+ D_j 2(\Sigma_j^{-1}(\mu_j - \mu_j^0)) \, . \end{aligned} \tag{4.8}$$

By setting them to zero, it implies,

$$\alpha_i + \beta_i = 1 \quad \forall i \tag{4.9}$$

and,

$$\mu_j = \Phi_j(\alpha, \beta) = \frac{\sum_i (\alpha_i - \beta_i) \sum_t \gamma_t^i(j) x_t + D_j \mu_j^0}{\sum_i (\alpha_i - \beta_i) \sum_t \gamma_t^i(j) + D_j} \, , \tag{4.10}$$

which is the GBW update equation for the mean vectors.

BW algorithm is a special case of GBW, since if we disable the regularization ($D = 0$) and set all α to one and β to zero for all references and $\alpha = \beta = 0.5$ for all competitors, we get

$$\mu_j = \frac{\sum_{i \in \mathsf{ref}} \sum_t \gamma_t^i(j) x_t}{\sum_{i \in \mathsf{ref}} \sum_t \gamma_t^i(j)} \,, \tag{4.11}$$

which is the BW update equation. EBW is also a special case of GBW, since if we set α equals one and β equals zero for all references, and α equals zero and β equals one for all competitors, the GBW update equation becomes EBW update equation,

$$\mu_j = \frac{\sum_{i \in \mathsf{ref}} \sum_t \gamma_t^i(j) x_t - \sum_{i \in \mathsf{com}} \sum_t \gamma_t^i(j) x_t + D_j \mu_j^0}{\sum_{i \in \mathsf{ref}} \sum_t \gamma_t^i(j) - \sum_{i \in \mathsf{com}} \sum_t \gamma_t^i(j) + D_j} \,. \tag{4.12}$$

One should note that this result implies the D-term used in the EBW algorithm can be considered as a regularization function using Mahalanobis distance between the mean vectors of the new and the backoff model, and the meaning is well represented.

If the optimization is performed on the covariance, the modification to the primal problem is

$$\min_{\epsilon, \Sigma} \sum_i \epsilon_i + \sum_j D_j (\mu_j^0 \Sigma_j^{-1} \mu_j^0 + \mathsf{tr}(\Sigma_j^0 \Sigma_j^{-1}) + \log |\Sigma_j|)$$

$$\begin{aligned} \text{s.t.} \quad & \epsilon_i \geq Q_i(\Sigma) - C_i \quad \forall i \\ & \epsilon_i \geq C_i - Q_i(\Sigma) \quad \forall i \,, \end{aligned} \tag{4.13}$$

Then, we have this Lagrangian, L_c^P,

$$\begin{aligned} L_c^P(\epsilon, \Sigma, \alpha, \beta) \;=\; & \sum_i \epsilon_i - \sum_i \alpha_i (\epsilon_i - Q_i(\Sigma) + C_i) \\ - \; & \sum_i \beta_i (\epsilon_i - C_i + Q_i(\Sigma)) \\ + \; & \sum_j D_j (\mu_j^0 \Sigma_j^{-1} \mu_j^0 + \mathsf{tr}(\Sigma_j^0 \Sigma_j^{-1}) \\ + \; & \log |\Sigma_j|) \,. \end{aligned} \tag{4.14}$$

We then differentiate the L_c^P w.r.t. the covariance,

$$\begin{aligned}\frac{\partial L_c^P}{\partial \Sigma_j} &= \sum_i (\alpha_i - \beta_i) \sum_t \gamma_t^i(j)(\Sigma_j^{-1} - \Sigma_j^{-1} S_{tj} \Sigma_j^{-1}) \\ &+ D_j(\Sigma_j^{-1} - \Sigma_j^{-1}\Sigma_j^0\Sigma_j^{-1} - \Sigma_j^{-1}\mu_j^0\mu_j^{0'}\Sigma_j^{-1}) \ , \end{aligned} \tag{4.15}$$

where $S_{tj} \equiv (x_t - \mu_j)(x_t - \mu_j)'$. Then by setting it to zero, we obtain the GBW update equation for covariance,

$$\begin{aligned}\Sigma_j = & \quad \Psi_j(\alpha, \beta) \\ = & \quad \frac{\sum_i (\alpha_i - \beta_i) \sum_t \gamma_t^i(j) x_t x_t' + D_j(\Sigma_j^0 + \mu_j^0 \mu_j^{0'})}{\sum_i (\alpha_i - \beta_i) \sum_t \gamma_t^i(j) + D_j} - \mu_j \mu_j' \ , \end{aligned} \tag{4.16}$$

which is also a generalization of BW and EBW. Instead of solving two independent optimization problems, one may use the $\{\alpha\}$ and $\{\beta\}$ obtained from the first problem as the solution for the second problem to compute the covariances. This procedure assumes the solutions of the two problems are similar and we adopt this procedure in our experiments. One should also note that the formulation of GBW can incorporate I-smoothing [Povey (2003)] similarly by adding another regularization term.

GBW is the same as BW and EBW that it is based on the EM algorithm. However, the M-step of GBW is now replaced by solving a dual problem to retrieve the Lagrange multipliers, so we can use equation 4.10 and equation 4.16 to obtain the HMM parameters. The dual problem is formulated by plugging equation 4.9, 4.10 and 4.16 into the Lagrangian. Assuming we are optimizing the mean vectors, we have

$$\begin{aligned} \max_{\alpha,\beta} \quad & L^D(\alpha, \beta) && = \sum_i (\alpha_i - \beta_i)(Q_i - C_i) \\ \text{s.t. } \forall i \quad & \alpha_i + \beta_i && = 1 \\ & \alpha_i, \beta_i && \geq 0 \ . \end{aligned}$$

This dual problem can be solved by gradient ascent. By taking derivative w.r.t. the Lagrange multipliers, we obtain the gradients.

$$\frac{\partial L^D}{\partial \alpha_i} = Q_i - C_i \ , \tag{4.17}$$

When α_i is updated, β_i can be obtained using the constraint $\alpha_i + \beta_i = 1$.

Finally, figure 4.1 summarizes the whole process of transforming the original optimization algorithm using Lagrange relaxation.

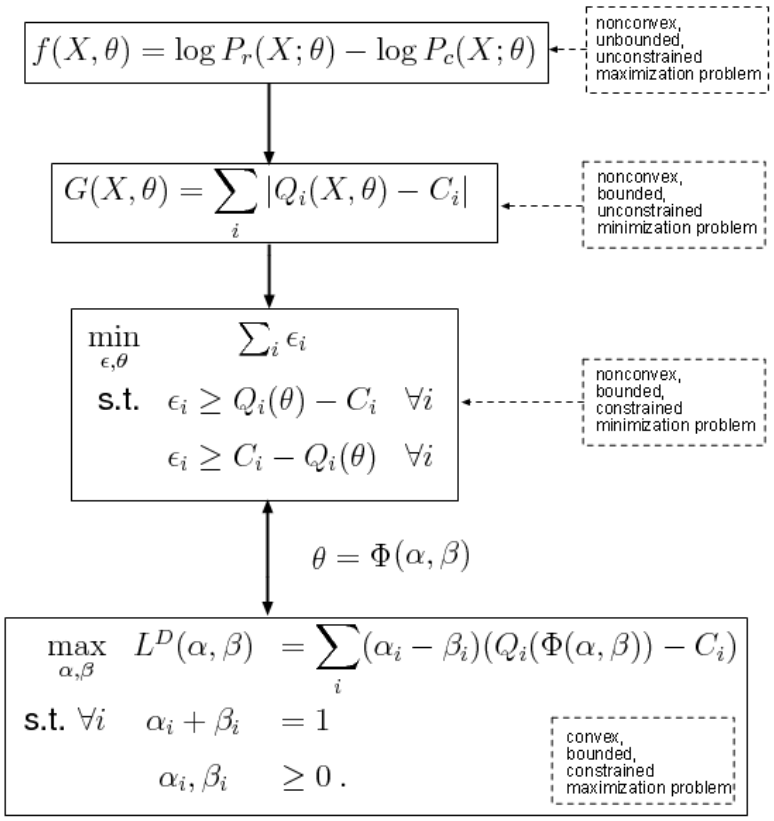

Figure 4.1: The process of transforming the problem using Lagrange relaxation.

4.4 GBW, EBW and Information Theory

We showed that EBW is a special case of GBW and the D-term in EBW can be expressed as some regularization to the optimization problem. In the past, this D_j constant is set by some heuristics, say $E \times \gamma_{den}$, which is tuned empirically and E is often set to some value

between one to two [Povey (2003)]. The formulation of GBW now justifies the heuristics that the researchers have been using from a theoretical point of view. Because from the optimization problem in equation 4.4 and equation 4.13, the regularization is only meaningful if the dynamic range of the regularization term is comparable to the transformed MMI objective function. In such a case, the D_j constant has to be proportional to the occupancy count, say γ_r or γ_c. Hence, GBW explains why a heuristic like $E \times \gamma_c$ is desirable to determine the values of D_j. In section 4.4.2, we discuss why γ_c is preferred over γ_r and also why E is preferred to be larger than or equal to one.

4.4.1 Recursive EBW/GBW Algorithm (rEBW/GBW)

The GBW algorithm also gives another interesting insight about the EBW algorithm. It states that the D-term in the EBW algorithm comes from some distance based regularization. In fact, GBW further explains that such regularization is based on a well known similarity measure between two probability distributions, i.e. KL divergence [Hsiao and Schultz (2011)].

If we combine the optimization problems for solving mean vectors and covariance matrices into one single problem, we have,

$$\min_{\epsilon,\mu,\Sigma} \sum_i \epsilon_i + \sum_j \frac{D_j}{2}(||\mu_j - \mu_j^0||_{\Sigma_j} + \mathsf{tr}(\Sigma_j^0 \Sigma_j^{-1}) + \log|\Sigma_j|)$$

$$\begin{aligned} \text{s.t.} \quad & \epsilon_i \geq Q_i(\mu,\Sigma) - C_i \quad \forall i \\ & \epsilon_i \geq C_i - Q_i(\mu,\Sigma) \quad \forall i \,. \end{aligned} \tag{4.18}$$

The regularization function is the KL-divergence from $N_0(\mu_j^0, \Sigma_j^0)$ to $N(\mu_j, \Sigma_j)$. If we put back the terms that are removed by differentiation,

$$\begin{aligned} \mathsf{KL}(N_0||N) &= \frac{1}{2}[||\mu_j - \mu_j^0||_{\Sigma_j} + \mathsf{tr}(\Sigma_j^0 \Sigma_j^{-1}) \\ &- \log\frac{|\Sigma_j^0|}{|\Sigma_j|} - D] \,, \end{aligned} \tag{4.19}$$

where D is the dimension of the feature vector. It is important to note that the term $\mu_j^0 \Sigma_j^{-1} \mu_j^0$ is moved from the mean optimization problem to the covariance optimization problem. This term is part of the Mahalanobis distance but it disappears when we differentiate the objective function with respect to the mean vectors, hence, it remains in the covariance problem as shown in equation 4.13.

Equation 4.18 and 4.19 show that the D-term in the EBW update equation comes from the KL-divergence. Without affecting the solution of the optimization problem, we use cross entropy as the regularization function,

$$\mathsf{CH}(N_0||N) = H(N_0) + \mathsf{KL}(N_0||N) \; . \tag{4.20}$$

This does not alter the solution because the entropy of the backoff Gaussian N_0,

$$H(N_0) = \frac{1}{2} \log((2\pi e)^D |\Sigma_0|) \; , \tag{4.21}$$

is not related to the mean and covariance that we are optimizing. The function $H(N_0)$ is derived from differential entropy and details are available in Ahmed and Gokhale (1989).

In this setting, cross entropy measures the average number of bits required to encode N given N_0 is the true distribution. This is reasonable for regularization since cross entropy increases when N moves too far away from the backoff Gaussian N_0. However, N_0 in the EBW algorithm is either the ML model or the model from the previous EM iteration. In most cases, N_0 is inferior and it is not the true distribution. While the true distribution is unknown, we can look for a better Gaussian for the backoff purpose.

In the first attempt, we suggest we can treat the EBW/GBW update equations as some recurrence relations. The M-step of the EBW algorithm becomes an iterative procedure,

$$\mu_j^{m+1} = \frac{\sum_i (\alpha_i - \beta_i) \sum_t \gamma_t^i(j) x_t + D_j \mu_j^m}{\sum_i (\alpha_i - \beta_i) \sum_t \gamma_t^i(j) + D_j} \; , \tag{4.22}$$

$$\Sigma_j^{m+1} = \frac{\sum_i (\alpha_i - \beta_i) \sum_t \gamma_t^i(j) x_t x_t' + D_j (\Sigma_j^m + \mu_j^m \mu_j^{m'})}{\sum_i (\alpha_i - \beta_i) \sum_t \gamma_t^i(j) + D_j} - \mu_j^{m+1} \mu_j^{m+1'} \; , \tag{4.23}$$

where μ_j^{m+1} and Σ_j^{m+1} are the Gaussian parameters of the $(m+1)$-th iteration, which depend on the parameters on the m-th iteration; If we perform only one iteration, it is the

same as the standard EBW/GBW algorithm. If we perform two iterations, it is like we are using the Gaussian computed from standard EBW/GBW algorithm as a backoff parameter. If we believe the Gaussian computed from the standard EBW/GBW algorithm is better than the original model, we are using a better estimate to compute the cross entropy for regularization. In this thesis, we use the variable M to denote how many M-steps are performed after each E-step.

The reason for choosing cross entropy instead of KL-divergence is to examine the convergence of this recurrence relation, and whether the recurrence update leads to a smaller cross entropy. One can compare the cross entropy of successive iterations since it is measured by the number of bits. KL-divergence is a relative measure and it cannot compare the results of different iterations. In our previous work in [Hsiao and Schultz (2011)], we do not know if equation 4.22 and equation 4.23 may converge, but we found that in our experiments, the cross entropy always decreases as the recursion continues, which implies the changes on the Gaussian parameters diminish across iterations. Details on this are available in section 4.6.1, and the convergence condition is available in section 4.4.2.

We would like to emphasize that the implementation of the above recurrence update equations is very simple. One can perform multiple M-steps in the standard EBW/GBW algorithm to achieve the same result. This incurs negligible extra computation since the M-step does not involve data processing. In this thesis, we focus on the effectiveness of this new EBW algorithm. Hence, we do not test the recursive GBW algorithm, but simply use GBW as a tool to derive this new recursive EBW algorithm.

4.4.2 Statistical EBW/GBW Algorithm (sEBW/GBW)

I would like to thank Nagesha Venki for his very useful input to the recursive EBW/GBW algorithm in section 4.4.1. The conversation with Venki leads to another EBW/GBW algorithm which reveals interesting properties of the optimization problem and the heuristics we have been using for discriminative training.

In the recursive EBW algorithm, the update equation for the means and the covariances become recursive equations which allow multiple updates using the same statistics

collected from the E-step. While the number of recursions performed for the recursive EBW algorithm is determined empirically, the recurrence equation can be solved analytically which implies there is a more systematical way to determine how to update the parameters. Consider the recursive mean update equation in equation 4.22,

$$\begin{aligned} \mu_j^{m+1} &= \frac{\sum_t \gamma_t^r(j)x_t - \sum_t \gamma_t^c(j)x_t + D_j\mu_j^m}{\sum_t \gamma_t^r(j) - \sum_t \gamma_t^r(j) + D_j} \\ &= K\mu_j^m + (1-K)\mu_j^N \end{aligned} \tag{4.24}$$

where

$$K = \frac{D_j}{\sum_t \gamma_t^r(j) - \sum_t \gamma_t^c(j) + D_j} \tag{4.25}$$

$$\mu_j^N = \frac{\sum_t \gamma_t^r(j)x_t - \sum_t \gamma_t^c(j)x_t}{\sum_t \gamma_t^r(j) - \sum_t \gamma_t^r(j)} . \tag{4.26}$$

It is important to note that μ_j^N is the solution of μ_j if we disable the regularization ($D_j = 0$). Then, we can solve the recurrence equation,

$$\begin{aligned} \mu_j^{m+1} &= K\mu_j^m + (1-K)\mu_j^N \\ \Rightarrow \mu_j^{m+1} - \mu_j^N &= K(\mu_j^m - \mu_j^N) \\ \Rightarrow \mu_{jd}^m &= \mu_{jd}^N + K^m \end{aligned} \tag{4.27}$$

where μ_{jd}^m is the d-th element in the mean vector μ_j^m.

Equation 4.27 implies that the ratio K plays an important role on the solution of the recursive equation. If $K > 1$, as $m \to \infty$, μ_j^m goes to infinity as well, which means there is no solution. However, if $0 \leq K \leq 1$, μ_j^m converges to μ_{jd}^N as $m \to \infty$, which implies as the recursion continues, the effect of the regularization diminishes. If $K < 0$, it is unable to predict the result of μ_j^m, however, in the context of discriminative training, the heuristics that are used to set the value of D_j would prevent $K < 0$. It is interesting to note that if the numerator count and the denominator count are equivalent, $K = 1$ as long as $D_j > 0$, which implies the solution must converge. However, this may be a rare case.

Assuming that $D_j = E\sum_t \gamma_t^c(j)$ and $E = 2$ which is a commonly used heuristics to set the value of D_j,

$$K = \frac{2\sum_t \gamma_t^c(j)}{\sum_t \gamma_t^r(j) + \sum_t \gamma_t^c(j)} . \tag{4.28}$$

In this case, $K > 1$ if and only if the numerator count is strictly smaller than the denominator count. As discussed in chapter 2, this is known to be true that if a Gaussian appears more often as the competitor, the optimization problem becomes minimum likelihood problem which is unbounded. In sum, this can be considered as another way to prove at some condition, the optimization problem would have a solution if regularization is disabled. Another important implication of this finding is that it explains why the standard EBW uses $D_j = E\sum_t \gamma_t^c(j)$ instead of $D_j = E\sum_t \gamma_t^r(j)$ for the heuristics. It is because if we use the numerator count to compute D_j, there is no guarantee that $K \geq 0$ which the solution may be diverged. To guarantee that $K \geq 0$, we need to use the denominator count and $E \geq 1$. This conclusion supports the heuristics we have been using for EBW, though such heuristics was determined empirically instead of being formulated mathematically.

The solution to the recursive equations inspires us that we can derive another EBW algorithm. As mentioned, the value of K plays an important role to determine whether a Gaussian needs regularization. In that case, one may use the value of K to classify the Gaussians into two category: one with regularization ($D_j > 0$) and one without ($D_j = 0$). The idea of this variant of EBW algorithm is, if the numerator count dominates, it means K is small and one can update the Gaussian more aggressively. However, if the denominator count dominates, it means K is large and one should update the Gaussian in a more conservative way. The value of K is computed for each Gaussian. Hence, this variant of the EBW algorithm would consider the numerator and denominator statistics to perform the update. Therefore, this EBW algorithm is named statistical EBW algorithm (sEBW). In general, sEBW would classify the Gaussians in the acoustic model into N classes sorted by the value of K. The classes with smaller K will perform more recursive updates which the classes with larger K will perform update less. The exact number of classes and the range of each class would need to be tuned empirically. This general form of sEBW algorithm, however, would need much tuning since we need to decide the number of classes and also the upper bound and the lower bound of K for each class. In this thesis, we focus on the simplest form which we only have two classes, one with regularization which is like the standard EBW, and another without regularization, which is equivalent to performing infinite numbers of recursions for rEBW.

4.5 Convergence Condition of EBW and GBW

The optimization technique we use for GBW is known as Lagrange relaxation [Boyd and Vandenberghe (2004)], since it converts a primal problem into a dual problem. In theory, the dual problem is always a convex problem (maximizing a linear objective function here) [Boyd and Vandenberghe (2004)]. Note that when strong duality does not hold, which means the optimal value of the dual can only serve as a strict lower bound to the primal objective, there is no guarantee that the solution obtained from the dual is primal optimal. We can only consider this technique as a relaxation method.

Consider when $D \rightarrow \infty$ and this term dominates the objective function, strong duality occurs and GBW is guaranteed to converge in this case. Although the solution is simply the backoff model, this behavior is the same as EBW. However, given a problem and a finite D, if the solution of GBW is equivalent to BW or EBW, it can be shown GBW is guaranteed to converge for this specific problem. One should also note that the D constant in GBW is related to the target values, C. If these target values are set more aggressively, that is very high likelihood for reference and very low likelihood for competing hypotheses, GBW is very likely to reduce to EBW (but it is possible to construct artificial cases that GBW does not reduce to EBW). However, in such a case, the ϵ of the primal problem becomes larger, and therefore, D has to be larger for regularization to be effective. Hence, although we claim GBW must converge when it reduces to EBW, this case is equivalent to saying GBW must converge when $D \rightarrow \infty$.

4.6 Experiments

We evaluated the performance of GBW, EBW, rEBW and sEBW on the Farsi, Iraqi and MSA ASR system. Detailed system description is available in chapter 3 and table 4.1 summarizes the systems used in the experiments.

	Farsi ASR	Iraqi ASR	MSA ASR
Train data	110 hr	450 hr	1100 hr
System type	SI, 1-pass	SA, 1-pass	SA, 3-pass
Vocab size	33K	62K	737K
Adaptation	None	Incremental	Batch
# Gaussians	112K	308K	867K
LM	3-gram	3-gram	4-gram

Table 4.1: Description of the Farsi, Iraqi and MSA ASR systems.

4.6.1 Experiments on GBW

We first compared the performance of GBW and EBW on the Farsi ASR system. MMI objective was chosen for optimization. The target values were selected based on the model used in E-step, and they were set to be 5% to 20% higher than the log likelihood of the references, and 5% to 20% lower of the competitors. In the M-step, we performed four iterations of gradient ascent to update the dual variables. From the dual variables, we then reestimated the Gaussian parameters. No regularization nor smoothing was used for GBW in this experiment.

The results in figure 4.2 show that GBW without regularization and smoothing can improve the baseline ML system. In this experiment, GBW optimizes for the MMI objective function. When the target values are close to the scores of the ML model, GBW obtains less improvement which is reasonable since the training is closer to the ML training in those settings. However, if the target values are set too aggressively, the training may not converge since regularization is disabled. In sum, this experiment verifies the basic framework of GBW of optimizing the models towards the target values even without using any regularization. On the contrary, EBW does not work when there is no regularization nor smoothing and it just corrupts the model.

When GBW is performed with regularization and smoothing, one can initialize the dual parameters such that GBW is the same as BW or EBW at the beginning. GBW

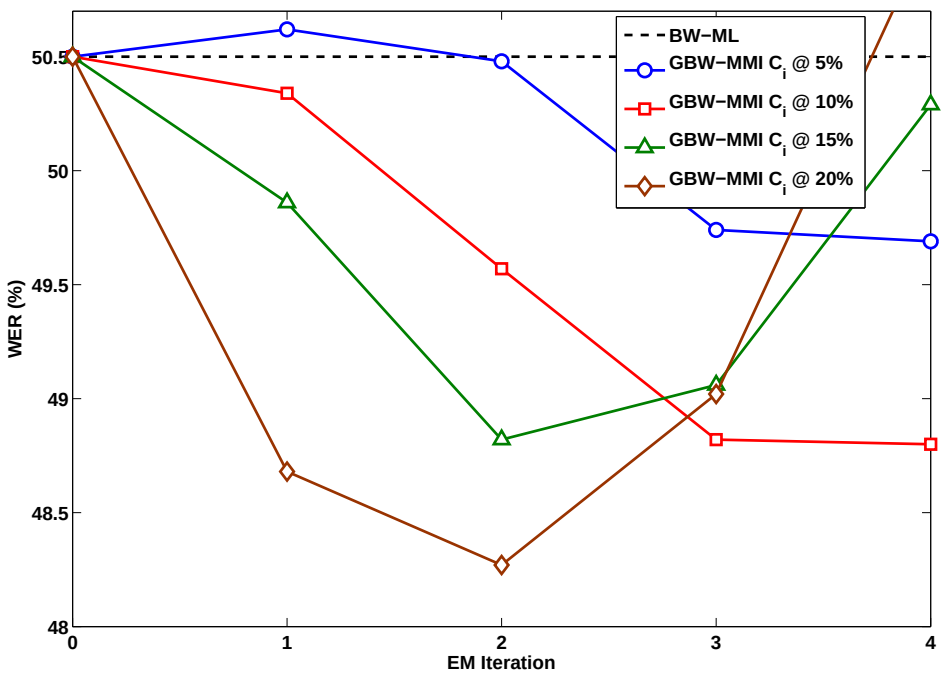

Figure 4.2: Performance of GBW without regularization on dev set. The percentage represents how far the target values are set based on the baseline model.

without regularization cannot be initialized as EBW since it may corrupt the model at the first iteration. One should note that although the dual problem is a convex problem and the initialization is not important, GBW is still under the EM framework and different starting points may yield different results. Another issue is when GBW is initialized as EBW, we have to first perform EBW for one iteration and use that model to perform E-step for the GBW at the beginning. This ensures the dual parameters match the Gaussian parameters of the model used in the E-step. It is always the case if we initialize GBW as BW because we use a ML model to perform E-step.

Figure 4.3 compares the performance of EBW and GBW with different initialization. In this experiment, regularization is enabled and the target values for GBW are always set to be 10% higher likelihood for references and 10% lower likelihood for the competi-

tors. The likelihood is computed using the ML model and the target values do not change during the optimization. As shown, when GBW is initialized as EBW, GBW has simi-

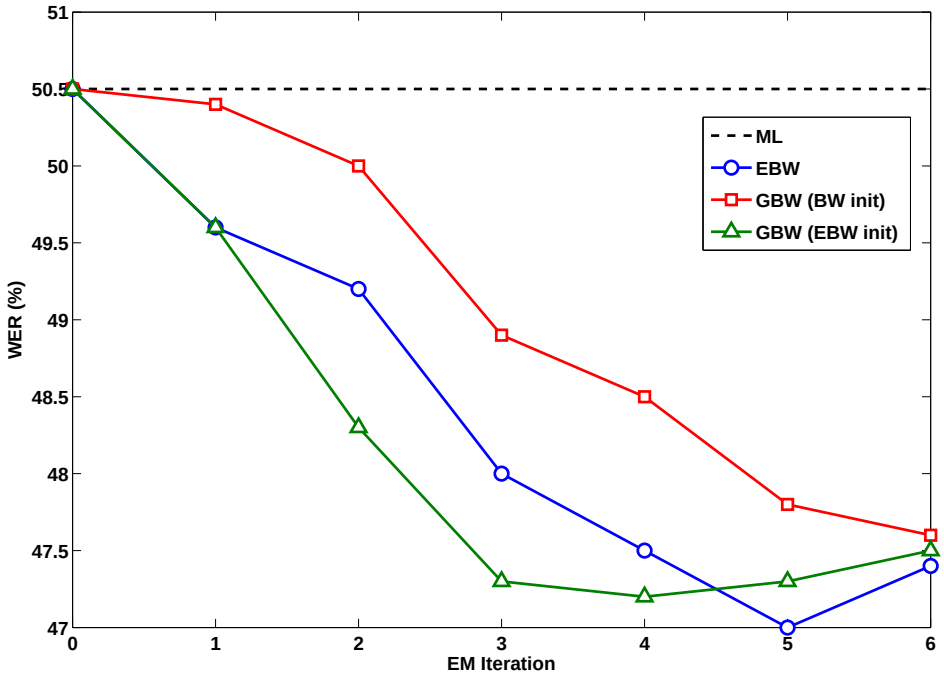

Figure 4.3: Performance of BW, EBW and GBW on TransTac Farsi July 2007 open set.

lar performance compared to EBW. GBW with EBW initialization achieves 47.2% WER while EBW reaches 47.0% WER. GBW with BW initialization lags behind EBW at the earlier stages of the training since GBW is close to ML at the beginning, but GBW can obtain the comparable performance of EBW at the end (47.6% WER).

In sum, GBW can perform model space discriminative training like EBW. The performance of GBW is comparable to EBW and GBW allows regularization to be disabled. The purpose of these experiments is to validate the formulas of GBW. While GBW does not have any practical advantage compared to EBW, we will see in the next section that the variants of EBW derived from GBW can improve the performance of discriminative training.

4.6.2 Experiments on EBW and rEBW

We then evaluated the performance of the proposed rEBW algorithm on the Farsi, Iraqi and MSA ASR system. Table 4.2 contains the time needed for each EM iteration of the EBW algorithm. The time was measured by using 20 cores running in parallel and each core had similar performance to the Intel Xeon X5355 series at 2.66GHz. It demonstrated discriminative training is very expensive. For the experiments, the Farsi system used the TransTac Jul07 Farsi open set as the unseen test set. The Iraqi system used the TransTac Jun08 open set as dev set, and Nov08 open set as the unseen test set. The MSA system used GALE dev07/08/09 as dev sets, and eval09 and a three hours subset of dev10 as the unseen test sets.

Farsi ASR	Iraqi ASR	MSA ASR
$\sim$2 hours	$\sim$12 hours	$\sim$5 days

Table 4.2: The time required for each EBW iteration on the Farsi, Iraqi and MSA ASR systems.

We first investigated how the recurrence update equations affect the performance of the new EBW algorithm. We compared the EBW algorithm with different number of M-steps per EM iteration using the recurrence equation 4.22 and 4.23. Both EBW algorithms optimize the acoustic model for the BMMI objective function. We used the Iraqi system to analyze the performance. In this experiment, we tried up to four EM iterations and for each EM iteration, we performed a fixed number of M-steps from one to four ($M = 1, 2, 3, 4$).

Figure 4.4 shows that if we perform more M-steps per EM iteration, the system can achieve the best performance at earlier iterations. However, as shown in figure 4.5, performing multiple M-steps may also cause overfitting to occur earlier than the standard EBW algorithm as the training becomes more aggressive. When we perform two M-steps per EM iteration ($M = 2$), we got 32.7% WER which is almost the same as the 32.6% WER of standard EBW ($M = 1$) with only half the training time. We also tried the standard EBW algorithm with a grid search of learning rate (E tuning). In the model update

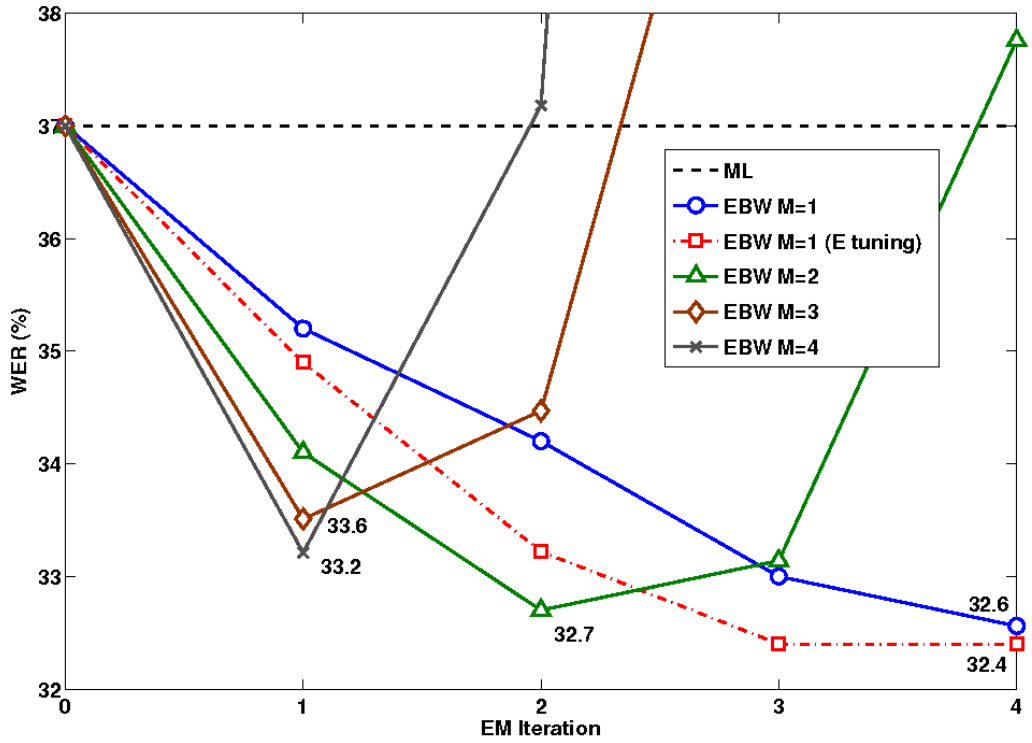

Figure 4.4: Performance of EBW algorithm with different number of M-steps per EM iteration. This experiment is performed on the TransTac Jun08 open set using the Iraqi ASR system.

equation 4.10 and 4.16, D_j controls the weight of the regularization. This value is often computed by a heuristics and it is the maximum of $E \times \sum_t \gamma_t^c(j)$, or twice the value required to keep the covariance positive. E is often set to two and it is also our setting for all EBW algorithms except the one with grid search. The grid search is performed based on the WER of the test set, which we find the best E in the range $[1.0, 3.0]$. Therefore, it is an oracle experiment. The purpose of this oracle experiment is to investigate if the standard EBW algorithm, in the optimal case, can converge as fast as our proposed EBW algorithm. Our results showed the opposite, and it implied our method is useful. Figure 4.6 shows the reduction in average cross entropy for each M-step performed. The cross entropy is computed after the first EM iteration shown in figure 4.4 and it is averaged across all Gaussian

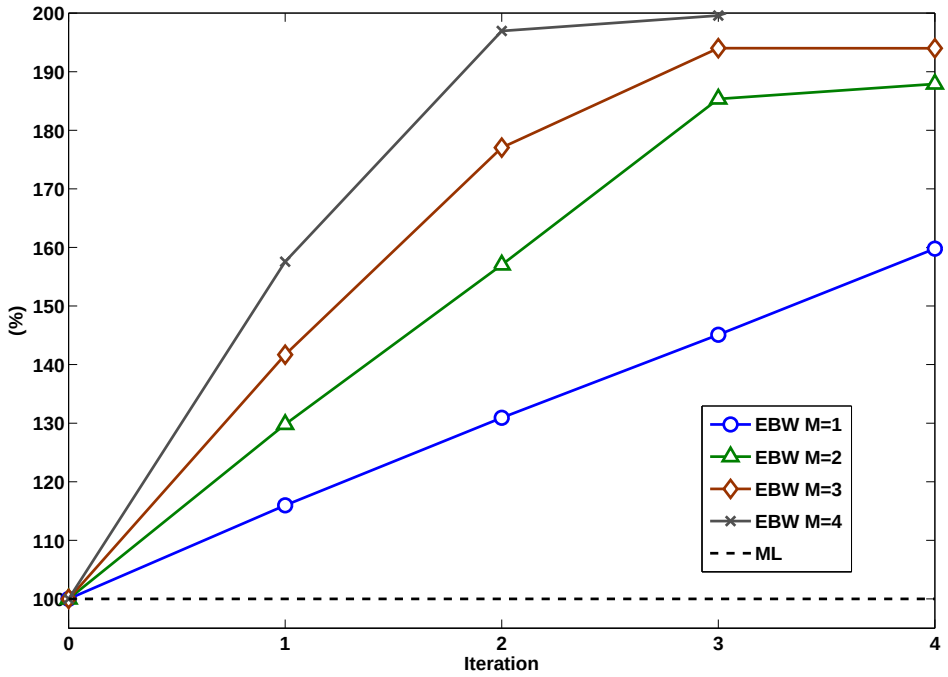

Figure 4.5: Increase of the BMMI objective function compared to the BMMI score of the ML model on the train set.

distributions in the acoustic model. This result shows that the cross entropy is decreasing so it implies the changes in the Gaussian parameters are also decreasing.

Based on these results, we studied whether our proposed rEBW algorithm causes accuracy degradation as a tradeoff for faster convergence. We compared the performance of the rEBW algorithm with the standard version on our Farsi ASR, Iraqi ASR and MSA ASR systems. In this experiment, the rEBW algorithm performed two M-steps for each E step ($M = 2$). In total, two EM iterations were performed. The standard EBW algorithm performed four EM iterations and one M-step per E-step ($M = 1$). Therefore, the execution time of the rEBW algorithm is only half of the standard version. Table 4.3, 6.4 and 6.5 showed the performance of the Farsi, Iraqi and MSA ASR systems respectively.

The results suggested that our proposed rEBW algorithm can achieve the same WER

	Farsi Jul07 open
BW_{ML}	50.2%
$EBW_{M=1}$	45.6%
$EBW_{M=2}$	45.5%

Table 4.3: The WER of the Farsi ASR system on the Jul07 open set.

	Jun08 open	Nov08 open
BW_{ML}	37.0%	35.2%
$EBW_{M=1}$	32.6%	30.6%
$EBW_{M=2}$	32.7%	30.8%

Table 4.4: The WER of the Iraqi ASR system on the Jun08 and Nov08 open sets.

	dev07	dev08	dev09	eval09	dev10
BW_{ML}	13.7%	15.5%	20.4%	15.1%	16.5%
$EBW_{M=1}$	11.7%	14.0%	18.6%	13.3%	14.6%
$EBW_{M=2}$	11.9%	14.0%	18.5%	13.2%	14.5%

Table 4.5: The WER of the MSA ASR system on the GALE dev07/08/09/10 and eval09 test sets.

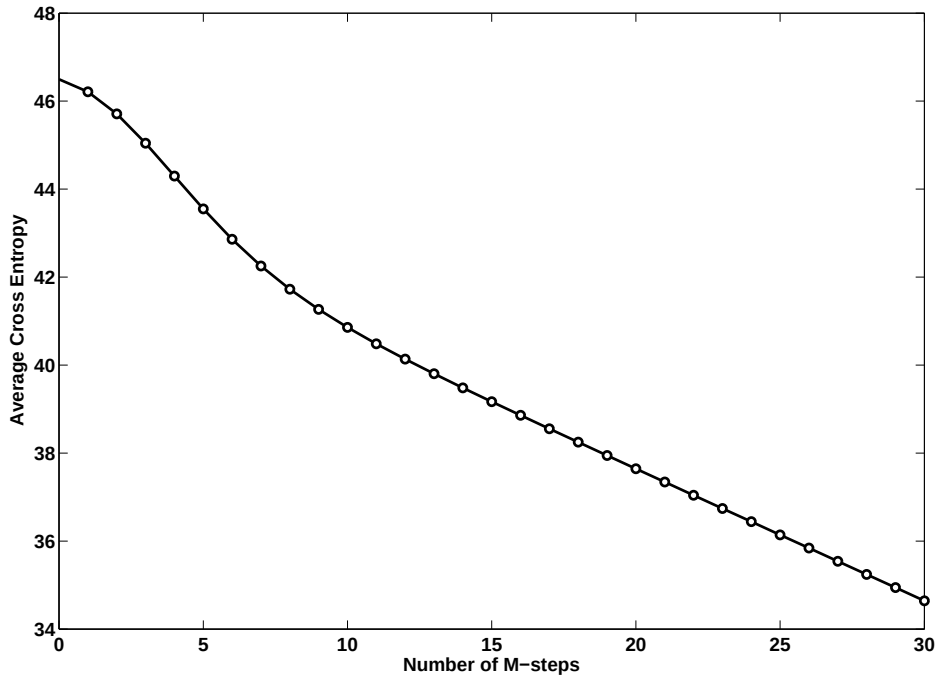

Figure 4.6: Decrease in average cross entropy implies the changes on the Gaussian parameters diminish for each M-step.

as the standard EBW algorithm. Among these eight test sets on three different systems, the difference in WER is no more than 0.2% absolute. Therefore, the gain in speed is a clear advantage of the rEBW algorithm. According to the information in table 4.2, using the standard EBW algorithm needs 20 days to train the MSA system, while the rEBW algorithm only needs around 10 days to achieve the same WER, which is a big advantage.

4.6.3 Experiments on EBW and sEBW

We then evaluated the performance of the sEBW algorithm also on the Iraqi and MSA ASR system. To apply the sEBW algorithm, one needs to classify the Gaussians into

different categories based on the ratio, K, where

$$K = \frac{D_j}{\sum_t \gamma_t^r(j) - \sum_t \gamma_t^c(j) + D_j} .$$

As described in section 4.4.2, if $0 \leq K \leq 1$, the Gaussian is in good condition that the solution of the optimization problem exists even if the regularization is disabled, otherwise, regularization is needed. However, the fact that a solution exists does not imply such solution is a good solution as it is still possible that the non-regularized solution may suffer from the overfitting issue. As a result, sEBW uses a threshold, L, such that as long as the ratio K of the Gaussian falls into the range $[0, L]$, regularization will be disabled for that Gaussian. If K is outside the range, we update the Gaussian like the standard EBW algorithm. The threshold L needs to be tuned empirically.

	L	# Gau w/ D=0	WER
EBW	0.0	0.0%	32.6%
sEBW	0.25	0.03%	32.5%
sEBW	0.375	0.13%	32.3%
sEBW	0.5	0.43%	32.3%
sEBW	0.75	3.68%	32.6%
sEBW	1.0	43.71%	39.8%

Table 4.6: WER of EBW, and sEBW on the TransTac Iraqi Jun08 open evaluation.

Table 4.6 shows the performance of EBW and sEBW on the Iraqi Jun08 test set with different thresholds. In this experiment, both EBW and sEBW optimize for the BMMI objective function. The result shows that sEBW algorithm can slightly improve the performance by choosing the threshold L properly. Although regularization is disabled for only very few Gaussians, it does affect the performance of the system.

On the unseen test sets, however, we do not see significant improvement as shown in table 4.7 for the Iraqi system and table 4.8 for the large scale MSA ASR system. On the MSA ASR system, we also tuned the threshold of L based on the dev sets and L was 0.5 in our experiments. It is not surprising to see little improvement since the heuristics for

	Jun08	Nov08
ML-BW	37.0%	35.2%
BMMI-EBW	32.6%	30.6%
BMMI-sEBW	32.3%	30.5%

Table 4.7: WER of EBW, and sEBW on the TransTac Iraqi test sets.

	dev07	dev09	eval09	dev10
ML-BW	13.7%	20.4%	15.1%	16.5%
BMMI-EBW	11.7%	18.6%	13.3%	14.6%
BMMI-sEBW	11.7%	18.4%	13.3%	14.6%

Table 4.8: WER of EBW and sEBW on the GALE MSA test sets.

discriminative training is tuned based on empirical approaches and it is believed to be near optimal.

Chapter 5

Generalized Discriminative Feature Transformation (GDFT)

5.1 Introduction

In this chapter, we introduce our proposed feature space discriminative training algorithm named generalized discriminative feature transformation (GDFT) [Hsiao and Schultz (2009), Hsiao et al. (2010)]. GDFT transforms the optimization problem of constrained maximum likelihood linear regression (CMLLR) [Digalakis et al. (1995); Gales (1998)] in a way similar to the GBW algorithm in chapter 4. The process of transforming the optimization problem is shown in figure 4.1 and it is also applicable to GDFT. Therefore, the transformed CMLLR can optimize for some discriminative objective function instead of likelihood. While CMLLR is a model space transformation technique originally designed for speaker adaptation, it can be shown CMLLR's transformation is equivalent to a feature space transformation [Gales (1998)]. Hence, CMLLR is also known as feature MLLR. In this chapter, we first review CMLLR and explain why it is equivalent to a feature transformation. Then, we show the formulation of GDFT and explain how it can be applied for feature space discriminative training.

5.2 CMLLR and Feature Transformation

CMLLR is a widely used speaker adaptation algorithm. CMLLR performs linear transformation on the Gaussian means and covariances, and restricts the transforms to be the same for mean and covariance. That is,

$$\hat{\mu} = H\mu - d \tag{5.1}$$

$$\hat{\Sigma} = H'\Sigma H \ , \tag{5.2}$$

where H and d are the rotation matrix and the bias to be optimized for likelihood respectively. For CMLLR, the auxiliary function is defined as,

$$\begin{aligned} Q(H,d) &= \sum_t \sum_j \gamma_t(j)[\log(|H'\Sigma_j H|) + (x_t - H\mu_j + d)'(H'\Sigma_j H)^{-1}(x_t - H\mu_j + d)] \\ &= \sum_t \sum_j \gamma_t(j)[\log(|H'\Sigma_j H|) + (x_t - H\mu_j + d)'H^{-1}\Sigma_j^{-1}H^{-1'}(x_t - H\mu_j + d)] \\ &= \sum_t \sum_j \gamma_t(j)[\log(|\Sigma_j|) + \log(|H|^2) \\ &+ (H^{-1'}x_t + H^{-1'}d - \mu_j)'\Sigma_j^{-1}(H^{-1'}x_t + H^{-1'}d - \mu_j)] \\ &= \sum_t \sum_j \gamma_t(j)[\log(|\Sigma_j|) - \log(|A|^2) + (W\zeta_t - \mu_j)'\Sigma_j^{-1}(W\zeta_t - \mu_j)] \ , \end{aligned} \tag{5.3}$$

where $A \equiv H^{-1'}$, $b \equiv H^{-1'}d$ and $W \equiv [A; b]$; $\zeta_t \equiv [x_t'; 1]'$ is the augmented feature vector. This formula is the auxiliary function or the negative log likelihood function for CMLLR. It also shows model transformation is equivalent to feature transformation as long as one subtracts $\log(|A|^2)$ to the log likelihood computation. This feature transformation is similar to how fMPE/MMI and RDLT transform the features except CMLLR optimizes for the likelihood instead of a discriminative objective function.

When context expansion and mean offset features are not used, the transformation matrix of fMMI/MPE is always square and identity, hence, fMMI/MPE can be considered as a model space transformation technique ($\log(|I|^2) = 0$). In contrast, RDLT is not a model space technique unless the likelihood computation is adjusted as CMLLR.

However, when there are more than one regression class, this conversion from model space to feature space may be more complicated. It depends on how the regression classes

are defined. The original CMLLR assigns transforms to the Gaussians [Digalakis et al. (1995)] and this assignment is predefined and fixed during adaptation. In such a case, each feature vector is needed to be transformed by different transforms depending on the Gaussian that we are evaluating. While this is complicated and inefficient, we propose to assign transforms based on the feature vectors like fMPE/MMI and RDLT. Given an incoming feature vector, we use a GMM to determine which transform we are going to use and update the features. This idea has been explored for CMLLR in [Kozat et al. (2006)] for speaker adaptation. The only difference is GDFT only allows one and only one transform to be assigned for each feature vector instead of a weighted sum using posterior probability. In section 5.5, we discuss in details why GDFT has such constraint. In any case, this scheme is equivalent to performing model transformation using a different transform for every feature vector. This is the approach we used for GDFT and later on, we will see this is very similar to fMPE/MMI and RDLT training.

We are interested in an approach similar to CMLLR, since as a model space technique, we have an option to update the transforms and the Gaussian parameters simultaneously, and it gives flexibility to the training procedure. If concurrent update of transformation parameters and Gaussian parameters is possible, it implies we can significantly reduce the total time for training. Also, we want the transformation optimized for an effective discriminative objective function like fMMI/MPE to improve recognition performance. In addition, we also want the transformation to be less restrictive like RDLT. Therefore, we propose GDFT as a feature space discriminative training algorithm.

5.3 GDFT and Lagrange Relaxation

Similar to the approach we used in the GBW algorithm (see figure 4.1), we first set up optimization problem for discriminative training on the linear transform, $W \equiv [A; b]$,

$$F(W) = Q_r(W) - Q_c(W) \ . \tag{5.4}$$

Minimizing F is the same as performing MMI optimization. However, optimization of F is not trivial since the solution can be unbounded. Instead of optimizing F directly, we

transform the problem as the GBW algorithm to limit the changes in the likelihood,

$$G(W) = \sum_i |Q_i(W) - C_i| \,, \tag{5.5}$$

where C_i is the chosen target value of Q as discussed in section 4.1.

Then, we show how to optimize equation 5.5. We would like to remind the readers that part of the formulation is closely related to CMLLR and readers are encouraged to read appendix C of [Gales (1998)] for more details. To minimize G, we first transform the problem to,

$$\begin{aligned} \min_{\epsilon,W} \quad & \textstyle\sum_i \epsilon_i \\ \text{s.t.} \quad & \epsilon_i \geq Q_i(W) - C_i \quad \forall i \\ & \epsilon_i \geq C_i - Q_i(W) \quad \forall i \,, \end{aligned} \tag{5.6}$$

where ϵ represents slack variables and i is an utterance index. This is equivalent to the original problem in equation 5.5 without constraints. We call this as the primal problem for GDFT .

We can then construct the Lagrangian dual for the primal problem. The Lagrangian is defined as,

$$\begin{aligned} L^P(\epsilon, W, \alpha, \beta) &= \sum_i \epsilon_i - \sum_i \alpha_i(\epsilon_i - Q_i(W) + C_i) \\ &- \sum_i \beta_i(\epsilon_i - C_i + Q_i(W)) \end{aligned} \tag{5.7}$$

where $\{\alpha_i\}$ and $\{\beta_i\}$ are the Lagrange multipliers for the first and the second set of constraints of the primal problem in equation 5.6. The Lagrangian dual is then defined as,

$$L^D(\alpha, \beta) = \inf_{\epsilon,W} L^P(\epsilon, W, \alpha, \beta) \tag{5.8}$$

Now, we can differentiate L^P w.r.t. ϵ and W which includes the rotation matrix A and the bias b. Hence,

$$\frac{\partial L^P}{\partial \epsilon_i} = 1 - \alpha_i - \beta_i \tag{5.9}$$

$$\frac{\partial L^P}{\partial W} = \sum_i (\alpha_i - \beta_i) \frac{\partial Q_i}{\partial W} \,. \tag{5.10}$$

By setting $\frac{\partial L^P}{\partial \epsilon_i}$ to zero, it implies,

$$\alpha_i + \beta_i = 1 \quad \forall i \ . \tag{5.11}$$

Assuming the covariance matrices are all diagonal, we then compute $\frac{\partial L^P}{\partial W}$ row by row,

$$\frac{\partial L^P}{\partial w_d} = \sum_i (\alpha_i - \beta_i) \frac{\partial Q_i}{\partial w_d} = -\Gamma \frac{p_d}{p_d w_d'} + w_d G^{(d)} - k^{(d)} \ , \tag{5.12}$$

where w_d refers to d-th row of W; $p_d = [c_{d1}, \ldots, c_{dn}, 0]$ is the extended cofactor row vector of A ($c_{ij} = cof(A_{ij})$) , and,

$$G^{(d)} = \sum_i (\alpha_i - \beta_i) \sum_j \frac{1}{\sigma_{jd}^2} \sum_t \gamma_t^i(j) \zeta_t \zeta_t' \tag{5.13}$$

$$k^{(d)} = \sum_i (\alpha_i - \beta_i) \sum_j \frac{1}{\sigma_{jd}^2} \mu_{jd} \sum_t \gamma_t^i(j) \zeta_t' \tag{5.14}$$

$$\Gamma = \sum_i (\alpha_i - \beta_i) \sum_t \sum_j \gamma_t^i(j) \ . \tag{5.15}$$

To solve $\frac{\partial L^P}{\partial w_d} = 0$, we can use the same method as CMLLR by first solving this quadratic equation for δ,

$$\delta^2 p_d G^{(d)-1} p_d' + \delta p_d G^{(d)-1} k^{(d)'} - \Gamma = 0 \ . \tag{5.16}$$

Then we can apply this update equation,

$$w_d = (\delta p_d + k^{(d)}) G^{(d)-1} \ . \tag{5.17}$$

Updating W is an iterative process like CMLLR since the cofactors depend on other rows. As a result, we need to apply equation 5.17 on the whole transformation several times and recompute the cofactors until it converges. It is important to note that GDFT reduces to CMLLR if $\alpha_i = 1$ and $\beta_i = 0$ for all references and $\alpha_i = \beta_i = 0.5$ for all competitors.

Equation 5.12 to 5.17 show how W can be computed if the Lagrange multipliers, α, β, are known. In other words, W in equation 5.17 is a function of α and β. To estimate the

multipliers, we need to construct the dual problem from the Lagrangian (equation 5.7). This can be done by integrating equation 5.11 and 5.17 into equation 5.7. Thus, we obtain,

$$L^D(\alpha,\beta) = \sum_i (\alpha_i - \beta_i)(Q_i(W^*) - C_i) \tag{5.18}$$

where W^* is a function of α and β computed by equation 5.17. Then, we can formulate the dual problem,

$$\begin{aligned} \max_{\alpha,\beta} \quad & L^D(\alpha,\beta) && = \sum_i (\alpha_i - \beta_i)(Q_i(W^*) - C_i) \\ \text{s.t. } \forall i \quad & \alpha_i + \beta_i && = 1 \\ & \alpha_i, \beta_i && \geq 0 \; . \end{aligned}$$

This dual problem is convex and it can be solved easily with gradient ascent. While the gradient formula can be complicated, the following approximation is good enough in general,

$$\frac{\partial L^D}{\partial \alpha_i} \simeq Q_i(W^*) - C_i \; . \tag{5.19}$$

Similar to GBW, GDFT does not fulfill the strong duality condition. As a result, the solution of the dual problem may not be primal optimal. One can only consider this approach as a relaxation approach which we relax a non-convex problem into a convex one.

5.3.1 Regularization for GDFT

In chapter 2 and 4, we discuss the importance of regularization and smoothing for model space discriminative training. The need of regularization during optimization is due to the objective functions. However, regularization is not thoroughly explored for feature space discriminative training. While there are many smoothing techniques or heuristics available for the EBW algorithm, there are not many techniques designed for fMPE/MMI or RDLT except a heuristics of setting the learning rate for gradient descent [Povey et al. (2005)].

In this section, we show how the formulation of GDFT can be extended to incorporate regularization and how this regularization can improve GDFT. Adding regularization to

feature transformation is not new. In [Saon et al. (2009)], a large margin based semi-tied covariance (STC) method is developed. In that algorithm, a regularization scheme similar to GDFT is proposed. What separated the approach by [Saon et al. (2009)] and GDFT is that the regularization of GDFT is based on a distance measure like GBW and it is made explicit in the optimization problem.

Assuming $W^0 \equiv [A^0; b^0]$ is the backoff transform that we want to regularize in the optimization of GDFT, we can modify the GDFT objective function info,

$$G(W) = \sum_i |Q_i(W) - C_i| + \frac{D}{2}||W - W^0||_F^2 \,, \tag{5.20}$$

and modify the GDFT primal problem in equation 5.6 into,

$$\begin{array}{ll} \min\limits_{\epsilon,W} & \sum_i \epsilon_i + \frac{D}{2}||W - W^0||_F^2 \\ \text{s.t.} & \epsilon_i \geq Q_i(W) - C_i \qquad \forall i \\ & \epsilon_i \geq C_i - Q_i(W) \qquad \forall i \,, \end{array} \tag{5.21}$$

where $||W - W^0||_F$ is the Frobenius norm between W and W^0 and D is a tuning parameter to control to significance of the regularization term. The Lagrangian then becomes,

$$\begin{aligned} L^P(\epsilon, W, \alpha, \beta) &= \sum_i \epsilon_i - \sum_i \alpha_i(\epsilon_i - Q_i(W) + C_i) - \sum_i \beta_i(\epsilon_i - C_i + Q_i(W)) \\ &+ \frac{D}{2}||W - W^0||_F^2 \\ &= \sum_i \epsilon_i - \sum_i \alpha_i(\epsilon_i - Q_i(W) + C_i) - \sum_i \beta_i(\epsilon_i - C_i + Q_i(W)) \\ &+ \frac{D}{2}\sum_{i,j}(W_{ij} - W_{ij}^0)^2 \\ &= \sum_i \epsilon_i - \sum_i \alpha_i(\epsilon_i - Q_i(W) + C_i) - \sum_i \beta_i(\epsilon_i - C_i + Q_i(W)) \\ &+ \frac{D}{2}\sum_d (w_d - w_d^0)(w_d - w_d^0)' \end{aligned} \tag{5.22}$$

where w_d and w_d^0 represent the d-th row of W and W^0 respectively.

Same as the original GDFT, we differentiate this Lagrangian w.r.t. ϵ and set it to zero. Then we obtain,

$$\alpha_i + \beta_i = 1 \quad \forall i \; . \tag{5.23}$$

For $\frac{\partial L^P}{\partial W}$, we need to assume all covariance matrices are diagonal like CMLLR or the original GDFT. We compute the partial derivative row by row,

$$\begin{aligned} \frac{\partial L^P}{\partial w_d} &= \sum_i (\alpha_i - \beta_i) \frac{\partial Q_i}{\partial w_d} + D(w_d - w_d^0) \\ &= -\Gamma \frac{p_d}{p_d w_d'} + w_d (G^{(d)} + DI) - (k^{(d)} + D w_d^0) \\ &= -\Gamma \frac{p_d}{p_d w_d'} + w_d \tilde{G}^{(d)} - \tilde{k}^{(d)} \; , \end{aligned} \tag{5.24}$$

where

$$\tilde{G}^{(d)} = G^{(d)} + DI \tag{5.25}$$

$$\tilde{k}^{(d)} = k^{(d)} + D w_d^0 \; . \tag{5.26}$$

After that, we can follow the rest of the equations in the original GDFT to solve for W.

This is an interesting finding since adding regularization to GDFT only requires little modification to GDFT. The only changes to the formulation are how we compute G and k. The additional computation is negligible. Also, by adding $D \times I$ to G, as long as D is large enough, it helps G to have enough rank for inversion and this also reduces possible numerical issues. There are many possible choices of W^0. The simplest case is the identity matrix, I. Other possible choices include ML estimates, MMI estimates or the transform from the previous iteration. The D parameter serves as the same purpose as the D-term used in the EBW and GBW algorithm and we apply the same heuristics, i.e. $D = E \times \gamma_c$. When there are more than one regression classes, we have one D value for each transform, which is like one D value for each Gaussian in EBW or GBW.

5.3.2 Context Training for GDFT

As described, GDFT performs linear transformation on the feature vectors directly. In contrast, fMMI/MPE and RDLT can exploit the information available in the features within

a context window, and also high dimensional posterior features. The linear transforms trained by fMMI/MPE and RDLT project the high dimensional features to the original feature space. The projection can be considered as some form of feature selection and it is optimized for some discriminative objective function. We propose an optimization algorithm for GDFT to perform a similar function, which allows GDFT to exploit the information available in different features.

Suppose we try to estimate a projection matrix P,

$$G(P) = \sum_i |Q_i(P) - C_i| + \frac{D}{2}||P - P^0||_F^2 \,, \tag{5.27}$$

where

$$Q_i(P) = \sum_t \sum_j \gamma_t^i(j)(Py_t - \mu_j)'\Sigma_j^{-1}(Py_t - \mu_j) \tag{5.28}$$

which is an auxiliary function to represent negative log likelihood; P^0 is the backoff projection. The projection matrix P projects the high dimensional feature y_t to the original feature space. y_t can be constructed using the original feature x_t. For example, $y_t = [x'_{t-F}, \ldots, x'_t, \ldots, x'_{t+F}, 1]'$ where y_t is a supervector constructed by stacking the features within a context window of $\pm F$ frames. While there are many different ways to construct y_t, this paper focuses on the context features.

Similar to GBW and GDFT, we use Lagrange relaxation to solve equation 5.27, First, we construct an equivalent constrained optimization problem,

$$\begin{array}{lll} \min\limits_{\epsilon,P} & \sum_i \epsilon_i & +\dfrac{D}{2}||P - P^0||_F^2 \\ \text{s.t.} & \epsilon_i \geq Q_i(P) - C_i & \forall i \\ & \epsilon_i \geq C_i - Q_i(P) & \forall i \,. \end{array} \tag{5.29}$$

Then, we can setup the Lagrangian,

$$\begin{aligned} L_P(\epsilon, \mu, \alpha, \beta) &= \sum_i \epsilon_i - \sum_i \alpha_i(\epsilon_i - Q_i(P) + C_i) \\ &- \sum_i \beta_i(\epsilon_i - C_i + Q_i(P)) \\ &+ \frac{D}{2}||P - P^0||_{\Sigma_j}^2 \end{aligned} \tag{5.30}$$

where $\{\alpha_i\}$ and $\{\beta_i\}$ are the Lagrange multipliers for the first and the second set of constraints of the optimization problem in equation 5.29.

Now, we can differentiate L_P w.r.t. P and ϵ, and set them to zero, it implies,

$$\alpha_i + \beta_i = 1 \quad \forall i \; . \tag{5.31}$$

For the projection, we need to solve the equation in a row-by-row manner,

$$\begin{aligned} \frac{\partial L_P}{\partial P_d} &= \sum_i (\alpha_i - \beta_i) \frac{\partial L_P}{\partial P_d} + D(P_d - P_d^0) \\ &= \sum_i (\alpha_i - \beta_i) \sum_t \sum_j \gamma_t^i(j) \frac{1}{\sigma_{jd}^2} (P_d y_t - \mu_{jd}) y_t \\ &+ D(P_d - P_d^0) \; . \end{aligned}$$

Finally, we obtain,

$$P_d = k_y^{(d)} G_y^{(d)-1} \tag{5.32}$$

where P_d is the d-th row of P, and,

$$G_y^{(d)} = \sum_i (\alpha_i - \beta_i) \sum_j \frac{1}{\sigma_{jd}^2} \sum_t \gamma_t^i(j) y_t y_t' + DI \tag{5.33}$$

$$k_y^{(d)} = \sum_i (\alpha_i - \beta_i) \sum_j \frac{\mu_{jd}}{\sigma_{jd}^2} \sum_t \gamma_t^i(j) y_t' + D P_d^0 \tag{5.34}$$

Similar to fMMI/MPE, the feature vectors are first transformed using the main transforms, W. Then, the features are stacked to form supervectors and we apply the projection as described in equation 5.32 to retrieve the final feature vectors in the feature space.

During training, the projection and the main transforms are optimized jointly. Although we can have multiple projections, we choose to have one projection transform and multiple main transforms like fMMI/MPE. For fMMI/MPE, only 10% of the training data is assigned to train the projection matrix. The reason is to prevent the projection simply scales the transformed features [Povey (2005)]. We adopt the same procedure for GDFT, which only 10% of the data is assigned to train the projection. In addition to solving the

issues mentioned in [Povey (2005)], this also greatly speeds up the training process since for 90% of the data, GDFT operates on the low dimensional features, as computing G^d and k^d are much more efficient than computing G_y^d and k_y^d. One should note that this procedure does not benefit fMMI/MPE in terms of computation since fMMI/MPE uses gradient descent and the computation of the gradient must involve the high dimensional features.

5.3.3 Training Procedure of GDFT

GDFT can be considered as a discriminative version of CMLLR. Although the transforms are applied on the features, GDFT can still be considered as a model space technique. The question is how GDFT should be integrated into the model training process.

One can consider GDFT as a feature transformation technique like fMPE/MMI and RDLT. In such a case, we can use the conventional approach which we first optimize the features. Once the features are optimized, we perform model space discriminative training to optimize the acoustic model. Another way to look at it is considering GDFT as a model space technique like CMLLR. One may first optimize the HMM parameters, then the model transforms. Or we may treat the transforms and the Gaussian parameters as a single parameter set and optimize them jointly. In sum, there are at least three possible training procedures.

GDFT is under the EM algorithm framework. While the E-step remains the same as CMLLR, the M-step is now replaced by solving a convex dual problem. To speed up the process, like GBW, we perform one iteration of gradient ascent in the dual problem to obtain the transforms, then we repeat another EM iteration.

5.4 Comparison on the Computational Complexity of GDFT and fMPE/MMI

Feature space discriminative training is known to be one of the most expensive process for discriminative training [Povey et al. (2005), Zhang et al. (2006a)]. The design of GDFT

considers computational cost as one of the most important factors and aims to achieve good recognition accuracy with low computational cost. In this section, we compare the computational cost of GDFT and fMMI.

There are two areas which we can compare the computational cost of GDFT and fMMI: the feature transformation process and the statistics accumulation process. For feature transformation, both GDFT and fMMI uses a GMM to assign the feature vectors to the feature transforms. The difference is GDFT assigns the feature to one and only one feature transform and fMMI generates the mean offset posterior features as discussed on section 2.3. For simplicity, we focus on the feature transformation part which GDFT transforms the features by,

$$y_t = A_i x_t + b_i \tag{5.35}$$

$$z_t = P y_t^F \tag{5.36}$$

and fMMI transforms the features by,

$$y_t^k = M_1^k h_t \tag{5.37}$$

$$z_{td} = x_{td} + \sum_{f=-F}^{F} \sum_{k} M_{2,(f+F,d)}^k y_{t+f}^k \ , \tag{5.38}$$

where y_t^F is the supervector of stacking $\pm F$ frames centered by y_t.

Computing y_t for GDFT needs $O(D^2)$ time where D is the dimension of the feature vector x_t, while fMMI needs $O(NKD^2)$ to compute all y_t^k where N is the number of Gaussians in the GMM and K is the number of blocks for block update. However, since h_t is sparse, so the actual computation for fMMI should be $O(AKD^2)$ where A is the average number of active Gaussians after the GMM evaluation.

For the final feature vector, z_t, GDFT needs to perform a projection using P to project y_t^F to D dimension. Therefore, GDFT needs $O(FD^2)$ time to compute z_t from y_t and the total time is also $O(FD^2)$. For fMMI, computing z_t from y_t needs $O(DFK)$, so the total time is $O(AKD^2 + FKD)$ which should be similar to GDFT's $O(FD^2)$ in normal configuration setup.

For the statistics accumulation process, GDFT needs to compute G^d, k^d, G_y^d and k_y^d as shown in equation 5.25, 5.26, 5.33, 5.34 respectively. Since the time for computing the G-matrices dominates the overall computation, we only need to consider the time for computing G. Recall that

$$G^d = \sum_i (\alpha_i - \beta_i) \sum_j \frac{1}{\sigma_{jd}^2} \sum_t \gamma_t^i(j) \zeta_t \zeta_t' + DI$$

One can precompute

$$T_{tjd}^i = (\alpha_i - \beta_i) \sum_j \frac{1}{\sigma_{jd}^2} \sum_t \gamma_t^i(j) \tag{5.39}$$

which costs $O(TJD)$ time where T is the length of the utterance and J is the number of Gaussians appeared in the reference and the competitor of utterance i. Then, once T_{tjd}^i is computed, we can accumulate G^d by,

$$G^d := G^d + \sum_t T_{tjd}^i \zeta_t \zeta_t' \,, \tag{5.40}$$

which costs $O(TD^2)$ time. As a result, the time for computing G^d is $O(TD^2 + TJD)$. However, there are D G-matrices, therefore, the total time for computing G is $O(TD^3 + TJD)$. It is not $O(TD^3 + TJD^2)$ because T_{tjd}^i does not need to be recomputed D times. For G_y^d, the way to compute the cost is the same except G_y^d is computed on y_t^F which is $2(F+1) \times D$ dimensional. As a result, the cost for computing G_y^d is $O(TF^2D^3 + TJD)$ where $\pm F$ is the width of the context window. In practice, the cost for GDFT is lower because only 10% of the data will be assigned to train the context transform, while for 90% of the data, GDFT operates on the low dimension features and has the complexity of $O(TD^3 + TJD)$. Also, the G-matrices are symmetric, therefore, one can further reduce the cost by half in practice.

For fMMI, the statistics accumulation process involves the computation of the gradients of the main transform, M_1 and the context transform. For simplicity, we ignore the costs of the extra BMMI pass for computing the indirect statistics and the ML update for

fMMI. Then, to compute the gradient of M_1,

$$\begin{aligned}\left(\frac{\partial Q_{\text{MMI}}^{\text{direct}}}{\partial M_1^k}\right)_{(i,j)} &= \sum_t \left(\frac{\partial Q_{\text{MMI}}^{\text{direct}}}{\partial z_t}\right)_i \sum_{f=-F}^{F} M_{2,(f+F,i)}^k h_{t+f,j} \\ \frac{\partial Q_{\text{MMI}}^{\text{direct}}}{\partial z_t} &= \sum_m \gamma_t^r(m)\Sigma_m^{-1}(z_t-\mu_m) - \sum_m \gamma_t^c(m)\Sigma_j^{-1}(z_t-\mu_m)\ .\end{aligned}$$

$\frac{\partial Q_{\text{MMI}}^{\text{direct}}}{\partial z_t}$ can be precomputed for the whole utterance and the total time is $O(TJD)$. For the rest, we can take the advantage that h_t is sparse, hence, we only need to accumulate the statistics as long as $h_{t+f,j}$ is non-zero. Then, the computing cost is $O(A \times 2(F+1) \times K \times D \times D) = O(AKFD^2)$. For M_2, computing the cost is straightforward,

$$\left(\frac{\partial Q_{\text{MMI}}^{\text{direct}}}{\partial M_2^k}\right)_{(f+F,i)} = \sum_t \left(\frac{\partial Q_{\text{MMI}}^{\text{direct}}}{\partial z_t}\right)_i y_{t+f,i}^k \tag{5.41}$$

so the total cost is $O(TAKFD + TJD)$. Same as GDFT, 10% of the data is assigned to train M_2 while for 90% of the data is assigned to train M_1.

For most of the data (90%), we compare $O(TD^3 + TJD)$ and $O(TAKFD^2 + TJD)$ for the runtime of GDFT and fMMI. Since D is around 40, it is going to be much smaller than $A \times K \times F$ for the setting suggested in [Povey (2005)] where $K = 9$ and $F = 8$. Therefore, GDFT runs faster than fMMI. It is important to note that the slowest case for GDFT which computes the statistics for the context transform ($O(TF^2D^3 + TJD)$) is slower than the average case for fMMI ($O(TAKFD^2 + TJD)$). Hence, as F increases, the speed advantage of GDFT would diminish.

5.5 Limitations of the GDFT Framework

Compared to fMPE/MMI and RDLT, GDFT has lower computational cost. One of the reasons comes from the CMLLR framework which has an efficient row-by-row update equations. However, this framework also imposes some constraint on GDFT that makes it difficult to adopt the posterior features or other high dimensional features used by fMPE/MMI or RDLT.

Suppose we would like to use the posterior features, which means instead of assigning one and only one transform for each frame, we assign multiple transforms weighted by the posterior probability distribution. Then, we would compute the transformed feature, y_t by

$$y_t = \sum_{n=1}^{N} \gamma_t(n)(A_n x_t + b_n) \tag{5.42}$$

where $\gamma_t(n)$ is the posterior probability at time t of n-th Gaussian in the GMM which is used by GDFT, fMPE/MMI and RDLT. However, this change would no longer fit into the CMLLR/GDFT framework which requires the transformation of the feature vectors is equivalent to the model transformation as shown in equation 5.1, 5.2 and 5.3.

If we relax the constraint and formulate GDFT as a pure feature transformation technique, we need to solve a similar but different optimization problem. For simplicity, we demonstrate the difficulties of solving the maximum likelihood problem if the features are transformed by equation 5.42. The reason is if problems occurs when solving $\frac{\partial Q}{\partial \theta} = 0$, the same problems will also affect the solution of solving $\sum_i (\alpha_i - \beta_i)\frac{\partial Q_i}{\partial \theta} = 0$. Consider the auxiliary function,

$$\begin{aligned} Q &= \sum_t \sum_j \gamma_t(j)(y_t - \mu_j)' \Sigma_j^{-1} (y_t - \mu_j) \\ &= \sum_t \sum_j \gamma_t(j)(\sum_{n=1}^{N} \gamma_t(n)(A_n x_t + b_n) - \mu_j)' \Sigma_j^{-1} (\sum_{n=1}^{N} \gamma_t(n)(A_n x_t + b_n) - \mu_j) \ , \end{aligned} \tag{5.43}$$

where minimizing Q is equivalent to maximizing the likelihood. Now, we differentiate Q with respect to A_n,

$$\frac{\partial Q}{\partial A_n} = \sum_t \sum_j \gamma_t(j) 2\Sigma_j^{-1} (\sum_{n=1}^{N} \gamma_t(n)(A_n x_t + b_n) - \mu_j) x_t' \tag{5.44}$$

and we need to solve $\frac{\partial Q}{\partial A_n} = 0$. For CMLLR or GDFT, we solve this problem row-by-row because by doing so, Σ_j^{-1} becomes a scalar if the model uses diagonal covariance. Therefore, we can move the parameter of interest, A, to the left hand side of the formula

and move the rest to the right hand side to derive the closed form solution. However, we can no longer do that in this scenario, since A_n is in the linear sum with the other transforms and we cannot derive a closed form solution like CMLLR or GDFT.

This problem does not imply $\frac{\partial Q}{\partial A_n} = 0$ cannot be solved analytically. As mentioned, although A_n is in the linear sum, one can still solve it by setting up a system of linear equations. However, the size of the linear system can be huge since we have $N \times D(D+1)$ parameters, it means the system has $N \times D(D+1)$ equations which would become intractable if we have more than a few hundred transforms.

A more practical way to solve the problem is to modify the formulation in section 5.3.2, which we derive the formulas to allow GDFT to perform context training. Instead of using equation 5.42 to perform feature transformation, we modify the equation into,

$$\begin{aligned} x_t^N &= [\gamma_t(1)x_t', \gamma_t(1), \gamma_t(2)x_t', \gamma_t(2), \ldots, \gamma_t(N)x_t', \gamma_t(N)]' && (5.45) \\ P &= [A_1; b_1; \ldots; A_N; b_N] && (5.46) \\ y_t &= Px_t^N \ , && (5.47) \end{aligned}$$

where x_t^N is a supervector constructed by stacking $[x_t', 1]' \times \gamma_t(n)$ for $n = 1, 2, \ldots, N$ and P is a D by $N \times D(D+1)$ projection matrix. In this case, the y_t computed by equation 5.47 is equivalent to the one computed by equation 5.42. However, the difference is we only have one projection matrix P to estimate instead of having multiple transforms. In this case, the auxiliary function becomes,

$$\begin{aligned} Q &= \sum_t \sum_j \gamma_t(j)(y_t - \mu_j)'\Sigma_j^{-1}(y_t - \mu_j) \\ &= \sum_t \sum_j \gamma_t(j)(Px_t^N - \mu_j)'\Sigma_j^{-1}(Px_t^N - \mu_j) \ . && (5.48) \end{aligned}$$

Then, we can differentiate Q with respect to P,

$$\frac{\partial Q}{\partial P} = \sum_t \sum_j \gamma_t(j) 2\Sigma_j^{-1}(Px_t^N - \mu_j)x_t^{N'} \ . \quad (5.49)$$

Finally, we need to solve $\frac{\partial Q}{\partial P} = 0$. To do so, we assume Σ_j is diagonal, then we can derive

a row-by-row update equation. Let P_d be the d-th row of P, then,

$$\begin{aligned}\frac{\partial Q}{\partial P_d} &= \sum_t \sum_j \gamma_t(j)\frac{1}{\sigma_j^2}(P_d x_t^N - \mu_{jd})x_t^{N'} = 0 \\ &\Rightarrow \; P_d(\sum_t \sum_j \gamma_t(j)\frac{1}{\sigma_j^2}x_t^N x_t^{N'}) = \sum_t \sum_j \gamma_t(j)\frac{\mu_{jd}}{\sigma_j^2}x_t^{N'} \; .\end{aligned}$$

As a result,

$$P_d = k_x^d G_x^{d-1} \tag{5.50}$$

where

$$G_x^d = \sum_t \sum_j \gamma_t(j)\frac{1}{\sigma_j^2}x_t^N x_t^{N'} \tag{5.51}$$

$$k_x^d = \sum_t \sum_j \gamma_t(j)\frac{\mu_{jd}}{\sigma_j^2}x_t^{N'} \; . \tag{5.52}$$

By transforming the problem like GBW and GDFT, we obtain the generalized version of the update equation,

$$P_d = k_g^d G_g^{d-1} \tag{5.53}$$

where

$$G_g^d = \sum_i (\alpha_i - \beta_i) \sum_t \sum_j \gamma_t^i(j)\frac{1}{\sigma_j^2}x_t^N x_t^{N'} \tag{5.54}$$

$$k_g^d = \sum_i (\alpha_i - \beta_i) \sum_t \sum_j \gamma_t^i(j)\frac{\mu_{jd}}{\sigma_j^2}x_t^{N'} \; . \tag{5.55}$$

This formulation is very flexible because we can have a closed form update equation regardless of how we construct the supervectors. However, the drawback is if the supervector has very high dimension, computing the G matrices is expensive. For context training, since only 10% of the data is applied, the cost is limited, but for posterior features, one has to collect the statistics over the whole train set which can be very expensive. In sum, while the framework allows GDFT to use posterior features or other high dimension features like fMPE/MMI and RDLT, the computational cost is expensive. As a result, we focus on the low dimension features and the context training for GDFT.

5.6 Experiments on GDFT

We evaluated the performance of GDFT on the Iraqi ASR system and the MSA ASR system. In the experiments, we study the how regularization and context training affects the performance of GDFT. We also compare the performance of GDFT and fMMI and also the combining the model space discriminative training. Table 5.1 summarizes the systems used in this section and detailed system description is available in chapter 3.

	Iraqi ASR	Unvow MSA ASR
Train data	450 hr	50 hr
System type	SA, 1-pass	SA, 2-pass
Vocab size	62K	737K
Adaptation	Incremental	Batch
# Gaussians	308K	52K
LM	3-gram	4-gram

Table 5.1: Description of the Iraqi and Unvow MSA ASR systems.

5.6.1 Experiments on GDFT about Regularization

Table 5.2 is the comparison of GDFT using different configurations using the Iraqi ASR system. The training in this experiment only consists of feature space training and the acoustic model is the ML model. For GDFT, the regularization parameter E is set from zero to two. From the results, we observed that regularization allows GDFT to use more transforms. The performance of GDFT without regularization degraded the accuracy when there were 1024 transforms and the training failed for 2048 transforms. However, with regularization, GDFT continued to improve the ML baseline with more than 1024 transforms. In this experiment, GDFT with regularization achieved 35.7% WER with 2048 transforms, which is better than the ML baseline with 1.3% absolute improvement.

In the experiment, we also explored how the regularization parameter, E, might affect

Training proc.	E	# transforms	WER
ML	-	-	37.0
GDFT	0.0	16	36.7
GDFT	0.0	1024	38.5
GDFT	0.0	2048	-
GDFT	1.0	16	36.7
GDFT	1.0	1024	36.1
GDFT	1.5	1024	36.2
GDFT	2.0	1024	36.2
GDFT	1.0	2048	35.7
GDFT	1.5	2048	35.8
GDFT	2.0	2048	35.9
GDFT	1.0	4096	35.9

Table 5.2: WER(%) of GDFT with and without regularization on the dev set (TransTac Jun08 open set).

the performance of GDFT. When there were 1024 transforms, GDFT with regularization had the same WER of 36.1% for different E from one to two. Similarly, when there were 2048 transforms, GDFT could outperform the baseline system with a WER of 35.7% for $E = 1$. From the results, we observed that the performance of GDFT was not sensitive to the choice of E which means tuning should be easy.

5.6.2 Experiments on GDFT about Context Training

In this experiment, we investigated how the size of the context window might affect the performance of GDFT. We used the unvowelized MSA 50-hr system to test different configurations.

Window	± 0	± 3	± 5	± 7	± 9
GDFT	18.9	18.9	18.6	18.5	18.8

Table 5.3: WER(%) of GDFT on the GALE dev07 test set for the Unvow 50-hr MSA system. The ML baseline is 19.8% WER.

Table 5.3 shows the performance of GDFT with context window of different size. In this experiment, GDFT optimized for the BMMI objective function and it used 1024 transforms. The acoustic model was the ML model. GDFT achieved the best performance when the window size was seven which concatenated ± 7 frames to construct the supervector and then projected it back to the original feature space using the context projection transform. The result showed that context training could improve the performance of GDFT. However, when model space discriminative training was applied on the acoustic model, the difference became smaller. In this particular system, GDFT without context training plus BMMI model space training gave 17.6% WER while GDFT with context training plus BMMI gave 17.4% WER.

5.6.3 Experiments on GDFT and fMMI

In section 5.4, we compare the computational complexity of GDFT and fMMI. While the analysis may help evaluating the efficiency of running GDFT and fMMI, it ignores the implementation details which may greatly affect the computational cost. Hence, we would like to perform some benchmarks to compare the actual runtime for GDFT and fMMI. We test GDFT and fMMI using different configurations and see how some configurations like the window size and block update may affect the speed and the recognition accuracy. For the experiment, we used the Iraqi ASR system to evaluate the algorithms. Both GDFT and fMMI had 2048 transforms. For simplicity, we only considered the gradient computation part for fMMI and ignored the costs for collecting indirect statistics and the ML model update. For timing, the benchmark was done by performing GDFT/fMMI on the whole Iraqi train set using 20 cores at $\sim$2.66GHz.

	window	block	pruned?	time/iter	WER
fMMI	$\pm$7	9	no	$\sim$5 days	35.2%
fMMI	$\pm$7	9	yes	$\sim$4 days	35.4%
fMMI	$\pm$7	1	yes	$\sim$2 days	35.7%
fMMI	$\pm$0	9	yes	$\sim$2 days	36.4%
fMMI	$\pm$0	1	yes	$\sim$1 day	36.5%
GDFT	$\pm$7	-	yes	$\sim$1 day	35.7%
GDFT	$\pm$0	-	yes	$\sim$0.5 day	35.8%

Table 5.4: Comparison on the runtime and the recognition performance on fMMI and GDFT. The WER is computed on the TransTac Iraqi Jun08 open set and the runtime is measured on the 450-hr train set.

Table 5.4 shows the runtime and the recognition performance of different configurations of GDFT and fMMI. In the table, the column "pruned?" means if yes, each frame can only be assigned to one and only one transform, which is required for GDFT. The default setting of fMMI on this parameter is no since fMMI uses posterior features, therefore, each frame can be assigned to multiple transforms depending on the posterior probability

distribution. The results in table 5.4 basically show that the posterior feature, the context expansion and the block update all contribute to the performance of fMMI. However, enabling all these features also increases the computational cost. In contrast, GDFT can achieve good performance if runtime is a concern. As shown in the experiment, GDFT can outperform fMMI in terms of recognition accuracy if we only allow at most one day per iteration. However, if we allow more training time, fMMI can outperform GDFT if all features are enabled.

	main transform	context transform
fMMI	$O(TAKFD^2 + TJD)$	$O(TAKFD + TJD)$
GDFT	$O(TD^3 + TJD)$	$O(TF^2D^3 + TJD)$

Table 5.5: Computation complexity of fMMI and GDFT for accumulating statistics.

In section 5.4, we analyze the computational complexity of GDFT and fMMI. Table 5.5 summarizes the complexity for accumulating the statistics for GDFT and fMMI. It is interesting to see that the runtime of GDFT with window size ± 7 is similar to the runtime of pruned fMMI with only context training or only block update. The reason is for some utterances, if the lattices are big, the runtime is dominated by the term $O(TJD)$, since J, which is the number of Gaussians appeared in the lattice, can be huge. In this case, the runtime of GDFT and fMMI can be similar. Another reason is when we compute the G matrices for GDFT, say equation 5.54, it can be computed by simply one function call using ATLAS BLAS [Whaley and Petitet (2005)]. ATLAS BLAS is a highly optimized libraries for linear algebraic operations, which helps lowering the computational cost of GDFT. In sum, GDFT has the advantage in terms of computational cost from both theoretical and implementation aspects compared to fMMI.

5.6.4 Experiments on Combining Model Space and Feature Space Discriminative Training

Then, we study the performance of combining feature space and model space discriminative training. For feature space discriminative training, we compare fMMI and GDFT, while for model space discriminative training, we use EBW and sEBW. The focus of these experiments are on the conventional training procedures, so we do not use rEBW for these experiments. The performance of fast discriminative training combining feature space and model space discriminative training will be addressed in chapter 6. In these experiments, both fMMI and GDFT used a context window of size ± 7 and fMMI had nine blocks of transforms. GDFT used the transforms from the previous EM iteration to perform backoff. All discriminative training algorithms reported in this section optimized the models for the BMMI objective function.

Figure 5.1 shows the performance of different training procedures using fMMI/GDFT and/or BMMI. The common strategy of combining feature space and model space discriminative training is first performing feature space discriminative training to optimize the features. Then, model space discriminative training is performed on the optimized features [Povey et al. (2005); Povey (2005)]. As shown in the figure, using feature space discriminative training such as GDFT or fMMI can improve the overall performance. It is interesting to see that although fMMI outperforms GDFT at the feature level, where GDFT gives 35.7% WER and fMMI gives 35.2% WER, the performance is almost the same after model space discriminative training where GDFT$\rightarrow$BMMI gives 31.9% WER and fMMI$\rightarrow$BMMI gives 31.8% WER. We also tried to combine GDFT, fMMI and BMMI together. In which, we performed GDFT first, then fMMI and finally BMMI. This is denoted by GDFT$\rightarrow$fMMI$\rightarrow$BMMI in figure 5.1 and the WER of this training procedure is 31.4% which slightly improves the procedures of using only GDFT or fMMI.

Table 5.6 shows the performance of using the sEBW algorithm as discussed in section 4.4.2 for the combined feature space and model space discriminative training procedure. In this experiment, the threshold for sEBW is tuned on the development set, i.e. TransTac Iraqi Jun08 open set, using the model space discriminative training only. Details

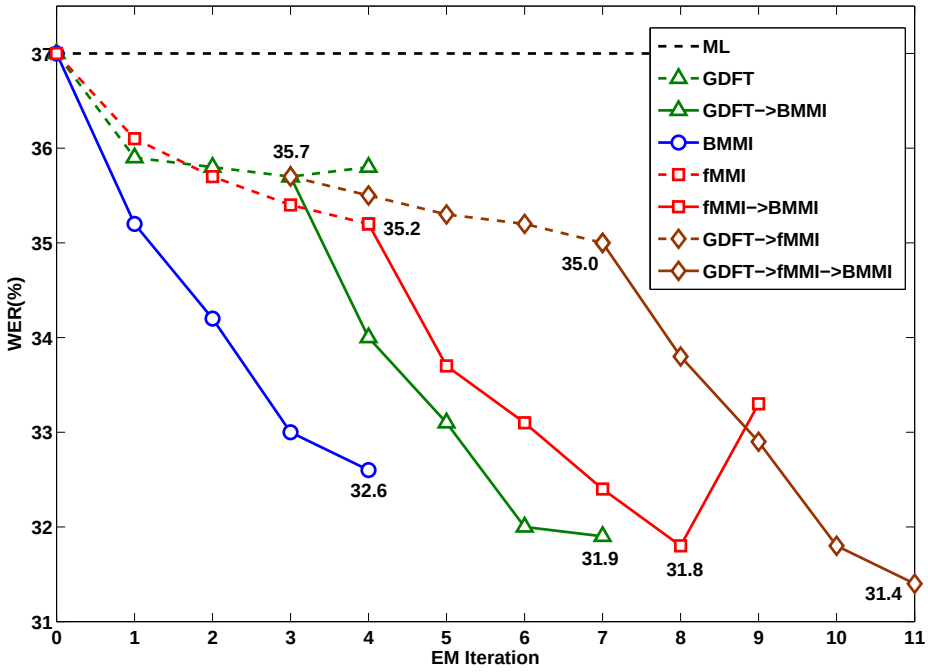

Figure 5.1: Performance of different ways to combine GDFT/fMMI with BMMI on the TransTac Iraqi Jun08 open set.

of the tuning process is available in section 4.6.3. From the results, although the sEBW algorithm does not significantly improve the recognition performance, the best system uses the sEBW algorithm combined with GDFT, fMMI and BMMI training. This system achieves 31.2% WER on the Jun08 open set and 29.8% WER on the unseen Nov08 open set.

	Jun08 open	Nov08 open
ML	35.7%	35.2%
BMMI-EBW	32.6%	30.6%
BMMI-sEBW	32.3%	30.5%
fMMI→BMMI-EBW	31.8%	30.0%
fMMI→BMMI-sEBW	31.6%	30.0%
GDFT→BMMI-EBW	31.9%	30.0%
GDFT→BMMI-sEBW	31.7%	30.0%
GDFT→fMMI→BMMI-EBW	31.4%	29.9%
GDFT→fMMI→BMMI-sEBW	**31.2%**	**29.8%**

Table 5.6: WER(%) of different discriminative training procedures and different EBW algorithms on the TransTac Iraqi Jun08/Nov08 open sets.

Chapter 6

Towards Single Pass Discriminative Training for Speech Recognition

6.1 Introduction

As discussed in previous chapters, discriminative training is an expensive but effective process to improve recognition accuracy for ASR systems. The lengthy training time is often due to the huge amount of data required to build a high performance system. Also, as long as the common belief – ”there is no data like more data” remains true, one can foresee that discriminative training will dominate the development time for an ASR system. This is not desirable for acoustic modeling research since discriminative training may greatly increase the turnaround time for experiments. Also, as tens of thousands of data become available nowadays, the cost of discriminative training may hinder the researchers to exploit the virtually unlimited amount of data to improve an ASR system.

In this chapter, we combine our proposed work and explore how much improvement we can achieve from discriminative training if we can only process the data once. If this single pass discriminative training is feasible, it helps lowering the cost of discriminative training.

6.2 Online Mode and Batch Mode for Discriminative Training

The idea of performing single pass discriminative training is not new. Researchers have been investigating single pass discriminative training in the form of online training. Instead of updating the model once after collecting the statistics from the whole train set, online training allows model update after processing each utterance. While online training is not limited to perform single pass training, it has good potential to perform single pass training since it allows more flexibility to adjust the model. Therefore, we would like to compare our proposed single pass discriminative training using our fast model space and feature space discriminative training algorithms with online discriminative training.

Existing online discriminative training algorithms are often based on stochastic gradient descent [Cheng et al.; Keshet et al. (2011)]. The reason is gradient can be computed for each utterance to perform model update. In contrast, Baum-Welch and Baum-Welch related algorithms require the statistics for the whole train set, which is not suitable for online training. However, Baum-Welch algorithms can be parallelized easily which is not the case for online gradient descent since if the model is updated every utterance, there is a sequential dependency which cannot be parallelized easily [Kuo et al. (2007)]. Therefore, batch training like Baum-Welch algorithm is often preferred when building a large scale system. As a result, batch mode training remains to be the main stream for building speech recognition systems.

Compared to batch training, it is more difficult to incorporate regularization for online training. As discussed in chapter 4, the performance of EBW and GBW heavily rely on the D-term which comes from the regularization function in the optimization problem, and the regularization considers the statistics of the whole train set. In stochastic gradient descent, it is not trivial to integrate such term as the overall statistics is not available. Existing online algorithms using gradient descent basically omit an explicit regularization function but rely on the objective function itself to prevent over train issues. This may also be the reason why MCE is more popular than BMMI or MPE for online training since MCE has a sigmoid function to control the over train issue [Cheng et al.; Keshet et al. (2011)].

In our work, we would like to take a different approach to the online training. Instead of using gradient descent, we would like to investigate if it is possible to use the EBW algorithm to perform online training. In chapter 4, we learn that the D-term comes from the regularization function. The question is if we can exploit the D-term, so that the batch update of the EBW algorithm would not be memoryless. If it works, we want to see how this version of online training perform in the case of single pass training. We also want to compare the online mode and the batch mode of single pass training and see how much improvement can be achieved by processing the data only once.

6.3 Experiments on Single Pass Discriminative Training

We conducted our experiments on two systems. Table 6.1 summaries the configuration of these systems. Detailed system description of the Iraqi ASR the MSA ASR system is available in chapter 3. For the experiments, the Iraqi system used the TransTac Jun08 open set as dev set, and Nov08 open set as the unseen test set. The MSA system used GALE dev07/09 as dev sets, and eval09 and a three hours subset of dev10 as the unseen test sets.

	Iraqi ASR	MSA ASR
Train data	450 hr	1100 hr
System type	SA, 1-pass	SA, 3-pass
Vocab size	62K	737K
Adaptation	Incremental	Batch
# Gaussians	308K	867K
LM	3-gram	4-gram

Table 6.1: Description of the Iraqi and the MSA ASR systems.

6.3.1 Experiments on Single Pass and Regular Discriminative Training

We first compared regular discriminative training procedures with the single pass discriminative training. For both regular and single pass training, we used BMMI for model space discriminative training, and for feature space training, we used fMMI and our proposed GDFT. Both fMMI and GDFT had a context window of ± 7 frames. Performing fMMI followed by BMMI training is considered to be the state of the art for discriminative training.

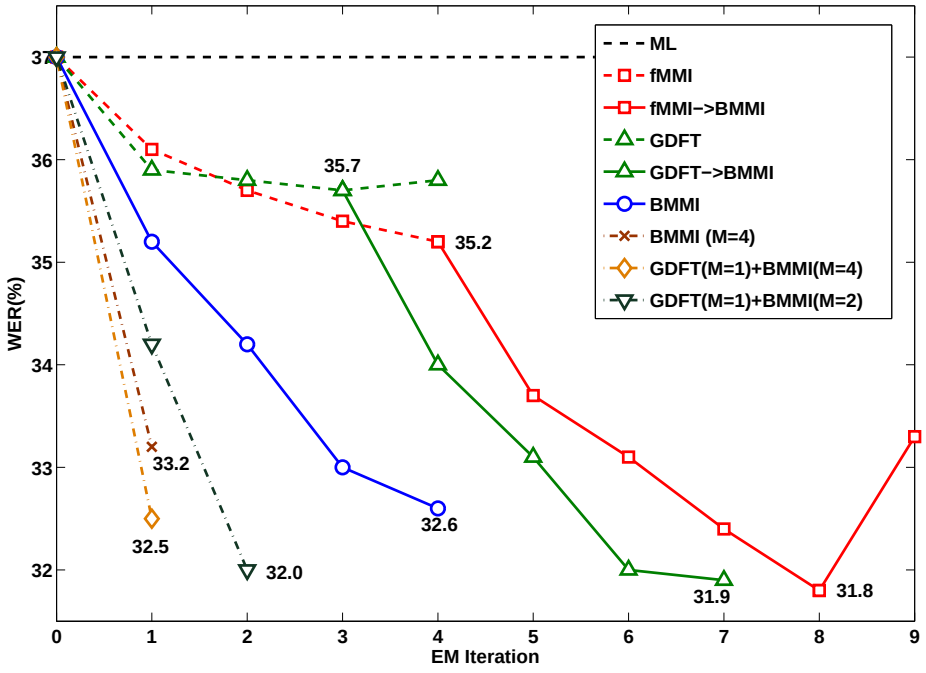

Figure 6.1: Performance of different training procedures. This experiment is performed on the TransTac Jun08 open set using the Iraqi ASR system.

Figure 6.1 shows that performing fMMI followed by BMMI (fMMI→BMMI) achieves 31.8% WER which improves the baseline ML model by 14.1% relative. If we replace

fMMI with GDFT, we get 31.9% WER which is very similar to fMMI→BMMI.

For single pass training, we achieve 32.5% WER by using one EM iteration of GDFT and one EM iteration of BMMI using the rEBW algorithm with four M-steps (M=4) per EM iteration. This performance is the same as the regular BMMI training without fMMI/GDFT (32.5%), but the regular BMMI training would need four passes on the train set instead of one. If we omit the GDFT for the single pass training, the performance is 33.2% WER. In sum, our single pass training achieves 86.5% of the total improvement available from discriminative training. If we release the single pass constraint and allow two passes of the data, GDFT(M=1)+BMMI(M=2) gives 32.0% WER at the second EM iteration. This means it obtains 96.1% of the improvement available in the best training procedure (fMMI→BMMI). Table 6.2 summarizes the cost of each EM iteration for GDFT, fMMI and BMMI. We can see that discriminative training is very expensive but our proposed training procedure can drastically reduce the computation and yet, obtain most of the improvement from discriminative training.

fMMI	GDFT	BMMI
∼5 days	∼1 day	∼12 hours

Table 6.2: The time required for each EM iteration of fMMI, GDFT and BMMI on the 450-hr Iraqi train set. The benchmark was done on 20 CPU cores @ ∼2.66GHz. For single pass training using GDFT and BMMI, the time is similar to running GDFT alone.

Model/GDFT	M=1	M=2	M=3	M=4
ML	35.9%	35.6%	35.7%	35.7%
BMMI(M=4)	32.5%	33.0%	33.1%	33.0%

Table 6.3: The performance of single pass training with different combination of M-steps for GDFT and BMMI. The experiment is performed on TransTac Jun08 Open set.

Although we only performed one M-step for each EM iteration for GDFT in the experiment shown in figure 6.1, we tried the recursive update for GDFT as well. Table 6.3

shows the results of using different ways to combine GDFT and BMMI for single pass training. When we use the ML model as the acoustic model, we observe GDFT can benefit from multiple M-steps. However, when we use the BMMI model (M=4) as the acoustic model, multiple M-steps for GDFT would degrade the performance. This result is reasonable since when we train the model and the feature transforms jointly using single pass training, BMMI is trained on the untransformed data, while GDFT assumes the acoustic model is the ML model. Hence, the mismatch becomes greater when we perform the recursive update. For single pass training, we found that the best setup is one M-step for GDFT and four M-steps for BMMI.

	#iters	Jun08open	Nov08open
ML	-	37.0%	35.2%
BMMI	4	32.6%	30.6%
fMMI→BMMI	4+4	31.8%	30.0%
GDFT→BMMI	4+4	31.9%	30.0%
BMMI(M=4)	1	33.2%	31.3%
GDFT(M=1) +BMMI(M=4)	1	32.5%	31.0%
GDFT(M=1) +BMMI(M=2)	2	32.0%	30.5%

Table 6.4: The WER of the Iraqi ASR system on the Jun08 and the unseen Nov08 open sets.

Table 6.4 and 6.5 show the performance of single pass discriminative training on the Iraqi and the MSA speech recognition systems for different test sets. These tables also show the number of EM iterations used for different training procedures. The time required for each EM iteration for different algorithms is available in table 6.2. In sum, the results are consistent with the first experiment, which single pass training using GDFT and the rEBW algorithm can achieve the performance of regular full BMMI training. If we allow two passes on the data, the performance of our proposed method is very close to the full fMMI and BMMI training.

	#iters	dev07	dev09	eval09	dev10
ML	-	13.7%	20.4%	15.1%	16.5%
BMMI	4	11.7%	18.6%	13.3%	14.6%
BMMI(M=4)	1	12.0%	18.6%	13.4%	14.7%
GDFT(M=1) +BMMI(M=4)	1	11.7%	18.5%	13.4%	14.6%

Table 6.5: The WER of the Vow 1100hrs 3-pass system on the GALE dev07/09/10 and eval09 test sets.

6.3.2 Experiments on Batch and Online Single Pass Discriminative Training

Then, we compare batch and online single pass training. Batch training means we update the model only after collecting the statistics from the whole train set. For online training, we allow model update after processing a subset of the data. One possible advantage of online training over batch training is the statistics collected by E-steps are computed by the model with better accuracy. While for batch training, the statistics are only collected by the ML model. To perform online training, we randomly splitted the train set of the Iraqi system into four subsets of equal size. A model update was performed after processing each subset. We repeated the experiments thrice with different data split and took the average WER as the results.

The results are shown in figure 6.2. In the figure, BMMI online 1 represents we perform BMMI with the standard EBW algorithm which we update the model (M=1) after processing each subset. BMMI online 1 has an WER of 33.4% after processing the whole train set. BMMI online 2 is similar to BMMI online 1 except the statistics accumulate and are not reset after processing each subset. Compared to BMMI online 1, BMMI online 2 has access to more data at the later stages of the training, but it may suffer from inconsistent statistics collected by the models at different stages. BMMI online 2 has an WER of 34.1% after the training. This result shows that the first strategy is better than the second

one. We also apply feature space discriminative training using GDFT to perform online training. GDFT+BMMI online uses the strategy of BMMI online 1 and it achieves 33.0% WER. However, both online training algorithms are worse than their corresponding batch mode training where BMMI using rEBW with four M-steps (M=4) has an WER of 33.2% while GDFT(M=1)+BMMI(M=4) has an WER of 32.5%.

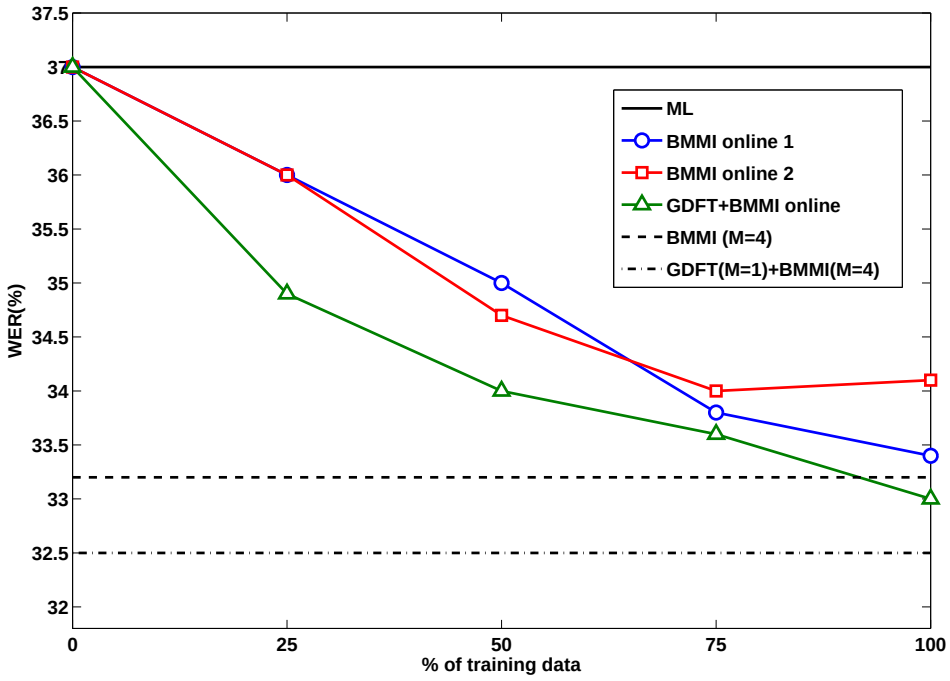

Figure 6.2: Performance of different online training procedures on TransTac Jun08 open set.

On the unseen TransTac Nov08 open set, we observe a similar trend at shown in table 6.6 which the batch mode single pass training is better than the online mode training. In sum, the batch training is slightly better than the online training in the context of single pass discriminative training.

	Jun08open	Nov08open
ML	37.0%	35.2%
BMMI(M=4)	33.2%	31.3%
BMMI(online)	33.4%	31.7%
GDFT(M=1) +BMMI(M=4)	32.5%	31.0%
GDFT +BMMI(online)	33.0%	31.6%

Table 6.6: Comparing the performance of the online and batch mode single pass discriminative training on the Iraqi Jun08 and the unseen Nov08 open sets.

Chapter 7

Conclusions

7.1 Contributions

We list out the contributions of our current work as follows:

- We have proposed the GBW algorithm which is the generalization of the BW and EBW algorithm. The formulation of GBW justifies the heuristics and the smoothing techniques used in the EBW algorithm from a theoretical aspect.
- The GBW framework also explains the EBW algorithm uses KL-divergence as a regularization function. This is a new insight which is not discovered in the original formulation of EBW. This finding also inspires better variants of the EBW algorithm. In our work, we propose the rEBW and the sEBW algorithms.
- Our proposed rEBW algorithm can reduce the time for model space discriminative training by half without any degradation on recognition accuracy. The rEBW algorithm can further speed up the training process up to four times faster with small degradation on accuracy.
- We have proposed the GDFT algorithm which generalizes CMLLR so it can perform feature space discriminative training. In our experiments, we have found that the

recognition performance of GDFT is comparable to fMMI but GDFT only requires around $\frac{1}{5}$ of the computation needed for fMPE/MMI.

- We have extended our previous work on GDFT so that it can perform context training like fMPE/MMI.

- By combining GDFT and rEBW, we have proposed single pass discriminative training which can achieve most of the improvement from discriminative training by only processing the data once. We also found that by allowing to process the data twice, we could achieve all of the improvement.

- We have compared our single pass discriminative training with online methods. While both methods only process the data once, single pass training gives better recognition accuracy.

- Our proposed optimization algorithms for discriminative training can greatly reduce the turnaround time for building an ASR system. This helps acoustic modeling research since to show one technique works, it is necessary to show that such technique can improve a discriminatively trained system. If the new technique is still under development, it can be very expensive to repeat the discriminative training process several times. By using GDFT, rEBW and single pass discriminative training, the turnaround time for an experiment would be greatly reduced, and hence, it helps the research on acoustic modeling.

- We have explained the meanings of indirect statistics used by fMPE/MMI and RDLT from the aspect of formulating an optimization problem. We showed that fMPE/MMI and RDLT optimize a multi-objective optimization problem. These insights help researchers to understand the feature space discriminative training algorithms better. These issues are not previously discussed in the original papers of fMPE/MMI and RDLT.

- The formulation of the GBW and GDFT algorithm shows one can convert many existing ML based algorithms to optimize for discriminative or other objective func-

tions. While this research only explored converting the BW and CMLLR algorithm, this formulation can be applied to many other algorithms as well.

7.2 Summary of Results

Using our baseline systems trained on sufficiently large amount of training data, we have achieved the following results:

Our experiments showed that the rEBW algorithm can achieve the same WER as the standard EBW algorithm. Among the eight test sets on three different systems, the difference in WER is no more than 0.2% absolute. The key advantage of rEBW is that it only requires half of the training time compared to the standard EBW algorithm. The rEBW algorithm can further speed up the training up to four times faster. However, that setting would slightly degrade the recognition accuracy by around $1-2\%$ relative compared to the standard EBW algorithm.

On the Iraqi ASR system, GDFT achieved $3-4\%$ relative reduction on WER which was not as good as fMMI which achieved around 5% relative reduction. However, after BMMI training, the difference was gone where both GDFT plus BMM and fMMI plus BMMI gave around $13-15\%$ relative WER reduction. However, GDFT only needs one fifth of the computation required for fMMI.

By combining all our proposed algorithms including GDFT and sEBW together with fMMI and BMMI, we achieved 31.2% and 29.8% WER for the Iraqi Jun08 and Nov08 open set which are the best numbers for this system in the thesis.

For the single pass discriminative training, we achieved $11-12\%$ relative WER reduction from discriminative training by processing the data only once. For single pass training, we jointly performed GDFT and BMMI using rEBW. Our single pass training is slightly better than the online training which gave around 10% relative reduction on WER.

7.3 Future Challenges and Potentials

Our proposed GBW algorithm is a new framework to formulate the optimization problem for discriminative training. This framework provides insights which are not discovered in the original formulation of the EBW algorithm. These insights help us to develop better variants of the EBW algorithm like our proposed rEBW and sEBW algorithms. While our focus in this thesis is to speed up discriminative training, the idea can be expanded to improve the recognition performance or perform semi-supervised training.

Under the GBW framework, we find that the EBW algorithm uses KL divergence as a regularization function but it assumes the model from the previous EM iteration as the true distribution. Our proposed rEBW algorithm exploits this by plugging in a discriminatively updated model for regularization. However, one can also plug in models estimated from or adapted for some unsupervised data. By doing so, we can come up with a new EBW algorithm which can combine supervised and unsupervised training. A similar idea has been explored in [Cui et al. (2011)] where the researchers proposed a variant of the EBW algorithm which aims to combine supervised and unsupervised discriminative training.

The formulation of GDFT context training provides a flexible framework to incorporate new features for GDFT. In the formulation, the projection is performed on the supervectors which construction is arbitrary. For context training, we stack features within a context window to form a supervector. However, one can construct the supervectors using posterior features or mean offset features like fMMI/MPE. The only drawback is the dimension of the supervectors cannot be too high or otherwise, the computational cost is expensive. This problem is avoided in context training because only a small portion of the data is assigned to train the projection. For other high dimensional features like posterior features, one may need to apply some dimension reduction methods like LDA before using GDFT.

The framework of GBW and GDFT using Lagrange relaxation can be applied to many speech problems. Our proposed formulation can transform an algorithm which originally optimizes for likelihood to optimize for some discriminative objective function. In addition, an explicit regularization function can be included in the optimization problem.

Generally speaking, if we have an ML algorithm which has an closed form solution for $\frac{\partial Q}{\partial \theta} = 0$ where Q is the auxiliary function, one can apply the same method used in GBW and GDFT to convert the ML algorithm to optimize for some discriminative objective function as long as we can solve $\sum_i (\alpha_i - \beta_i) \frac{\partial Q_i}{\partial \theta} = 0$, which is almost the same as the original problem except we have an additional factor $(\alpha_i - \beta_i)$. In most cases, we can derive a generalized version of the algorithm which can optimize for likelihood or some discriminative objective functions.

Bibliography

N.A. Ahmed and D.V. Gokhale. Entropy Expressions and their Estimators for Multivariate Distributions. *IEEE Transactions on Information Theory,* 35:688–692, 1989.

L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains. *Annals of Mathematical Statistics,* 41(1):164–171, 1970.

A. Biem and S. Katagiri. Feature Extraction based on Minimum Classification Error/Generalized Probabilistic Descent Method. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing,* 1993.

A. Biem and S. Katagiri. Filter bank design based on discriminative feature extraction. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing,* pages 485–488, 1994.

A. Biem, S. Katagiri, E. McDermott, and B. H. Juang. An Application of Discriminative Feature Extraction to Filter-Bank-Based Speech Recognition. *IEEE Transactions on Speech and Audio Processing,* 9(2):96–110, Feb 2001.

A. W. Black, K. Lenzo, and V. Pagel. Issues in Building General Letter to Sound Rules. In *Proceedings of The 3rd ESCA Workshop on Speech Synthesis,* pages 77–80, Australia, 1998.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

C. Cheng, F. Sha, and L. K. Saul. A Fast Online Algorithm for Large Margin Training of Continuous Density Hidden Markov Models. In *Proceedings of INTERSPEECH*, pages 668–671.

X. Cui, J. Huang, and J. T. Chien. Multi-view and Multi-objective Semi-supervised Learning for Large Vocabulary Continuous Speech Recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 4668 – 4671, 2011.

V.V. Digalakis, D. Rtischev, and L.G. Neumeyer. Speaker Adaptation Using Constrained Estimation of Gaussian Mixtures. *IEEE Transactions on Speech and Audio Processing*, 3:357–366, 1995.

M. J. F. Gales. Maximum Likelihood Linear Transformations for HMM-based Speech Recognition. *Computer Speech and Language*, 12:75–98, 1998.

M. J. F. Gales. Semi-Tied Covariance Matrices for Hidden Markov Models. *IEEE Transactions on Speech and Audio Processing*, 2, May 1999.

P. Gopalakrishnan, D. Kanevsky, A. Nadas, and D.Nahamoo. A Generalization of the Baum Algorithm to Rational Objective Functions. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 631–634, 1989.

P. Gopalakrishnan, D. Kanevsky, A. Nadas, and D.Nahamoo. An Inequality for Rational Functions with Applications to Some Statistical Estimation Problems. *IEEE Transactions on Information Theory*, 37(1):107–113, 1991.

A. Gunawardana and W. Byrne. Discriminative Speaker Adaptation with Conditional Maximum Likelihood Linear Regression. In *Proceedings of the European Conference on Speech Communication and Technology*, 2001.

X. He and L. Deng. *Discriminative Learning for Speech Recognition: Theory and Practice*. Morgan and Claypool Publishers, 2008.

R. Hsiao and T. Schultz. Generalized Baum-Welch Algorithm and Its Implication to a New Extended Baum-Welch Algorithm. In *Proceedings of INTERSPEECH*, 2011.

R. Hsiao and T. Schultz. Generalized Discriminative Feature Transformation for Speech Recognition. In *Proceedings of INTERSPEECH*, pages 664–667, 2009.

R. Hsiao, Y. C. Tam, and T. Schultz. Generalized Baum-Welch Algorithm for Discriminative Training on Large Vocabulary Continuous Speech Recognition System. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 3769–3772, 2009.

R. Hsiao, F. Metze, and T. Schultz. Improvements to Generalized Discriminative Feature Transformation for Speech Recognition. In *Proceedings of INTERSPEECH*, pages 1361–1364, 2010.

Q. Jin and T. Schultz. Speaker Segmentation and Clustering in Meetings. In *Proceedings of the International Conference on Spoken Language Processing*, 2004.

B.H. Juang and S. Katagiri. Discriminative Learning for Minimum Error Classification. *IEEE Transactions on Signal Processing*, 40(12):3043–3054, 1992a.

B.H. Juang and S. Katagiri. Discriminative Learning for Minimum Error Classification. *IEEE Transactions on Signal Processing*, 40(12):3043–3054, 1992b.

J. Keshet, C. C. Cheng, M. Stoehr, D. McAllester, and L. K. Saul. Direct Error Rate Minimization of Hidden Markov Models. In *Proceedings of INTERSPEECH*, 2011.

S. S. Kozat, K. Visweswariah, and R. A. Gopinath. Feature Adaptation based on Gaussian Posteriors. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 221–224, 2006.

H. K. J. Kuo, B. Kingsbury, and G. Zweig. Discriminative Training of Decoding Graphs for Large Vocabulary Continuous Speech Recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 4, pages 45–48, 2007.

L. Lee and R. Rose. Speaker Normalization Using Efficient Frequency Warping Procedures. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 353–356, Atlanta, 1996.

C.J. Leggetter and P.C. Woodland. Maximum Likelihood Linear Regressopn for Speaker Adaptation of Continuous Density Hidden Markov Models. *Computer Speech and Language*, 9:171–185, 1995.

R. G. Leonard. A Database for Speaker-Independent Digits Recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1984.

B. Mak, Y. C. Tam, and Q. Li. Discriminative Auditory Features for Robust Speech Recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 381–384, Orlando, Florida, USA, 2002.

B. Mak, Y. C. Tam, and R. Hsiao. Discriminative Training of Auditory Filters of Different Shapes for Robust Speech Recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 2, pages 45–48, Hong Kong, 2003.

M. Noamany, T. Schaaf, and T. Schultz. Advances in the CMU/InterACT Arabic GALE transcription system. In *Proceedings of HLT/NAACL*, 2007.

Y. Normandin and S. D. Morgera. An Improved MMIE Training Algorithm for Speaker-independent, Small Vocabulary, Continuous Speech Recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1991.

D. Pearce and H. Hirsch. The aurora experimental framework for the performance evaluation of speech recognition systems under noisy conditions. In *ISCA ITRW ASR2000*, pages 29–32, 2000.

D. Povey. Improvements to fMPE for Discriminative Training of Features. In *Proceedings of INTERSPEECH*, pages 2977–2980, 2005.

D. Povey. *Discriminative Training for Large Vocabulary Speech Recognition*. PhD thesis, Cambridge University Engineering Dept., 2003.

D. Povey and P. C. Woodland. Improved Discriminative Training Techniques for Large Vocabulary Continuous Speech Recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2001.

D. Povey and P. C. Woodland. Minimum Phone Error and I-smoothing for Improved Discriminative Training. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2002.

D. Povey, B. Kingsbury, L. Mangu, G. Saon, H. Soltau, and Geoffrey Zweig. fMPE: Discriminatively Trained Features for Speech Recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2005.

D. Povey, D. Kanevsky, B. Kingsbury, B. Ramabhadran, G. Saon, and K. Visweswariah. Boosted MMI for Model and Feature-space Discriminative Training. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 4057–4060, 2008.

G. Saon, D. Povey, and H. Soltau. Large Margin Semi-tied Covariance Transforms for Discriminative Training. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 3753–3756, 2009.

R. Schluter, W. Macherey, S. Kanthak, H. Ney, and L. Welling. Comparison of Optimization Methods for Discriminative Training Criteria. In *Proceedings of the European Conference on Speech Communication and Technology*, pages 15–18, 1997.

V. Valtchev, J. J. Odell, P. C. Woodland, and S. J. Young. MMIE Training of Large Vocabulary Recognition Systems. *Speech Communication*, 22(4):303–314, 1997.

L. Wang and P. C. Woodland. MPE-Based Discriminative Linear Transform for Speaker Adaptation. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 321–324, 2004.

L. R. Welch. Hidden Markov Models and the Baum-Welch Algorithm. *IEEE Information Theory Society Newsletter*, 54(4), 2003.

R. C. Whaley and A. Petitet. Minimizing Development and Maintenance Costs in Supporting Persistently Optimized BLAS. *Software: Practice and Experience*, 35(2):101–121, February 2005.

P. C. Woodland and D. Povey. Large Scale Discriminative Training For Speech Recognition. In *Proceedings of ISCA ITRW ASR2000*, pages 7–16, Paris, 2000.

K. Yu, M. J. F. Gales, and P.C. Woodland. Unsupervised Training with Directed Manual Transcription for Recognising Mandarin Broadcast Audio. In *Proceedings of INTERSPEECH*, 2007.

B. Zhang, S. Matsoukas, and R. Schwartz. Discriminatively Trained Region Dependent Feature Transforms For Speech Recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2006a.

B. Zhang, S. Matsoukas, and R. Schwartz. Recent Progress on the Discriminative Region-dependent Transform for Speech Feature Extraction. In *Proceedings of INTERSPEECH*, pages 1573–1576, 2006b.